

Package ‘msa’

May 9, 2024

Type Package

Title Multiple Sequence Alignment

Version 1.37.0

Date 2024-04-26

Author Enrico Bonatesta [aut], Christoph Kainrath [aut], Ulrich Bodenhofer [aut,cre]

Maintainer Ulrich Bodenhofer <ulrich@bodenhofer.com>

Description The ‘msa’ package provides a unified R/Bioconductor interface to the multiple sequence alignment algorithms ClustalW, ClustalOmega, and Muscle. All three algorithms are integrated in the package, therefore, they do not depend on any external software tools and are available for all major platforms. The multiple sequence alignment algorithms are complemented by a function for pretty-printing multiple sequence alignments using the LaTeX package TeXshade.

URL <https://github.com/UBod/msa>

License GPL (>= 2)

Copyright See file inst/COPYRIGHT

Depends R (>= 3.3.0), methods, Biostrings (>= 2.40.0)

Imports Rcpp (>= 0.11.1), BiocGenerics, IRanges (>= 1.20.0), S4Vectors, tools

Suggests Biobase, knitr, seqinr, ape (>= 5.1), phangorn, pwalgn

LinkingTo Rcpp

SystemRequirements GNU make

VignetteBuilder knitr

LazyLoad yes

Collate AllClasses.R AllGenerics.R params-methods.R version-methods.R helperFunctions.R inputChecks.R convertRows.R msaPrettyPrint.R print-methods.R show-methods.R msa.R msaMuscle.R msaClustalW.R msaClustalOmega.R msaConvert.R msaCheckNames.R msaConsensusSequence-methods.R msaConservationScore-methods.R

biocViews MultipleSequenceAlignment, Alignment, MultipleComparison, Sequencing

NeedsCompilation yes

git_url <https://git.bioconductor.org/packages/msa>

git_branch devel

git_last_commit dc8b00f

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-08

Contents

msa-package	2
msa	4
msaCheckNames	7
msaClustalOmega	8
msaClustalW	10
msaConsensusSequence	13
msaConservationScore	15
msaConvert	17
MsaMetaData-class	19
MsaMultipleAnlignmentClasses	20
msaMuscle	23
msaPrettyPrint	25

Index **30**

msa-package	<i>Multiple Sequence Alignment</i>
-------------	------------------------------------

Description

The **msa** package provides a unified R/Bioconductor interface to different multiple sequence alignment algorithms. Currently, ‘ClustalW’, ‘ClustalOmega’, and ‘MUSCLE’ are supported. All algorithms are usable without additional software packages and on all major platforms. The multiple sequence algorithms are complemented by an R interface to the powerful LaTeX package **texshade.sty** which allows for a highly customizable plots of multiple sequence alignments.

Details

The central method of this package is [msa](#). It provides unified access to all three multiple alignment algorithms implemented in this package. The function (like the three functions directly accessing the respective algorithm, i.e., [msaClustalW](#), [msaClustalOmega](#), and [msaMuscle](#)) can handle multiple types of input sequences, such as [XStringSet](#) and any subclass thereof, character vectors, and can also directly read from FASTA files. The package defines special classes for storing multiple

alignments and provides several methods for manipulating, analyzing, and printing such multiple alignments. The function `msaPrettyPrint` provides an interface to the powerful TeXshade package that allows for powerful LaTeX-based visualization of multiple alignments.

Author(s)

Enrico Bonatesta, Christoph Kainrath, and Ulrich Bodenhofer

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**(22):4673-4680. DOI: [doi:10.1093/nar/22.22.4673](https://doi.org/10.1093/nar/22.22.4673).

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Soeding, J., Thompson, J. D., and Higgins, D. G. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.* **7**:539. DOI: [doi:10.1038/msb.2011.75](https://doi.org/10.1038/msb.2011.75).

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**(5):1792-1797. DOI: [doi:10.1093/nar/gkh340](https://doi.org/10.1093/nar/gkh340).

Edgar, R. C. (2004). MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* **5**:113. DOI: [doi:10.1186/147121055113](https://doi.org/10.1186/147121055113).

Beitz, E. (2000). TeXshade: shading and labeling of multiple sequence alignments using LaTeX2e. *Bioinformatics* **16**(2):135-139. DOI: [doi:10.1093/bioinformatics/16.2.135](https://doi.org/10.1093/bioinformatics/16.2.135).

See Also

[msa](#), [msaClustalW](#), [msaClustalOmega](#), [msaMuscle](#), [msaPrettyPrint](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## call unified interface msa() for default method (ClustalW) and
## default parameters
msa(mySeqs)
```

Description

The `msa` function provides a unified interface to the three multiple sequence alignment algorithms in this package: ‘ClustalW’, ‘ClustalOmega’, and ‘MUSCLE’.

Usage

```
msa(inputSeqs, method=c("ClustalW", "ClustalOmega", "Muscle"),
    cluster="default", gapOpening="default",
    gapExtension="default", maxiters="default",
    substitutionMatrix="default", type="default",
    order=c("aligned", "input"), verbose=FALSE, help=FALSE,
    ...)
```

Arguments

<code>inputSeqs</code>	input sequences; this argument can be a character vector, an object of class <code>XStringSet</code> (includes the classes <code>AAStringSet</code> , <code>DNAStrngSet</code> , and <code>RNAStrngSet</code>), or a single character string with a file name. In the latter case, the file name is required to have the suffix ‘.fa’ or ‘.fasta’, and the file must be in FASTA format.
<code>method</code>	specifies the multiple sequence alignment to be used; currently, “ClustalW”, “ClustalOmega”, and “Muscle” are supported.
<code>cluster</code>	parameter related to sequence clustering; its interpretation and default value depends on the method; see <code>msaClustalW</code> , <code>msaClustalOmega</code> , or <code>msaMuscle</code> for algorithm-specific information.
<code>gapOpening</code>	gap opening penalty; the defaults are specific to the algorithm (see <code>msaClustalW</code> , and <code>msaMuscle</code>). Note that the sign of this parameter is ignored. The sign is automatically adjusted such that the called algorithm penalizes gaps instead of rewarding them.
<code>gapExtension</code>	gap extension penalty; the defaults are specific to the algorithm (see <code>msaClustalW</code> , and <code>msaMuscle</code>). Note that the sign of this parameter is ignored. The sign is automatically adjusted such that the called algorithm penalizes gaps instead of rewarding them.
<code>maxiters</code>	maximum number of iterations; its interpretation and default value depends on the method; see <code>msaClustalW</code> , <code>msaClustalOmega</code> , or <code>msaMuscle</code> for algorithm-specific information.
<code>substitutionMatrix</code>	substitution matrix for scoring matches and mismatches; format and defaults depend on the algorithm; see <code>msaClustalW</code> , <code>msaClustalOmega</code> , or <code>msaMuscle</code> for algorithm-specific information.

type	type of the input sequences <code>inputSeqs</code> ; possible values are "dna", "rna", or "protein". In the original ClustalW implementation, this parameter is also called <code>-type</code> ; "auto" is also possible in the original ClustalW, but, in this package, "auto" is deactivated. The type argument is mandatory if <code>inputSeqs</code> is a character vector or the file name of a FASTA file (see above). If <code>inputSeqs</code> is an object of class <code>AAStringSet</code> , <code>DNAStrngSet</code> , or <code>RNStringSet</code> , the type of sequences is determined by the class of <code>inputSeqs</code> and the type parameter is not necessary. If it is nevertheless specified and the type does not match the class of <code>inputSeqs</code> , the function stops with an error.
order	how the sequences should be ordered in the output object; if "aligned" is chosen, the sequences are ordered in the way the multiple sequence alignment algorithm orders them. If "input" is chosen, the sequences in the output object are ordered in the same way as the input sequences. For MUSCLE, the choice "input" is not available for sequence data that is read directly from a FASTA file. Even if sequences are supplied directly via R, the sequences must have unique names, otherwise the input order cannot be recovered. If the sequences do not have names or if the names are not unique, the <code>msaMuscle</code> function assigns generic unique names "Seq1"-Seqn to the sequences and issues a warning.
verbose	if TRUE, the algorithm displays detailed information and progress messages.
help	if TRUE, information about algorithm-specific parameters is displayed. In this case, no multiple sequence alignment is performed and the function quits after displaying the additional help information.
...	all other parameters are passed on to the multiple sequence algorithm, i.e. to one of the functions <code>msaClustalW</code> , <code>msaClustalOmega</code> , or <code>msaMuscle</code> . An overview of parameters that are available for the chosen method is shown when calling <code>msa</code> with <code>help=TRUE</code> . For more details, see also the documentation of chosen multiple sequence alignment algorithm.

Details

`msa` is a simple wrapper function that unifies the interfaces of the three functions `msaClustalW`, `msaClustalOmega`, and `msaMuscle`. Which function is called, is controlled by the method argument.

Note that the input sequences may be reordered by the multiple sequence alignment algorithms in order to group together similar sequences (see also description of argument `order` above). So, if the input order should be preserved or if the input order should be recovered later, we strongly recommend to always assign unique names to the input sequences. As noted in the description of the `inputSeqs` argument above, all functions, `msa()`, `msaClustalW`, `msaClustalOmega`, and `msaMuscle`, also allow for direct reading from FASTA files. This is mainly for the reason of memory efficiency if the sequence data set is very large. Otherwise, we want to encourage users to first read the sequences into the R workspace. If sequences are read from a FASTA file directly, the order of output sequences is completely under the control of the respective algorithm and does not allow for checking whether the sequences are named uniquely in the FASTA file. The preservation of the input order works also for sequence data read from a FASTA file, but only for ClustalW and ClustalOmega; MUSCLE does not support this (see also argument `order` above and `msaMuscle`).

Value

Depending on the type of sequences for which it was called, `msa` returns a [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), or [MsaRNAMultipleAlignment](#) object. If called with `help=TRUE`, `msa` returns an invisible `NULL`.

Author(s)

Enrico Bonatesta and Christoph Kainrath

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). `msa`: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

http://www.clustal.org/download/clustalw_help.txt

<http://www.clustal.org/omega/README>

<http://www.drive5.com/muscle/muscle.html>

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**(22):4673-4680. DOI: [doi:10.1093/nar/22.22.4673](https://doi.org/10.1093/nar/22.22.4673).

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Soeding, J., Thompson, J. D., and Higgins, D. G. (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.* **7**:539. DOI: [doi:10.1038/msb.2011.75](https://doi.org/10.1038/msb.2011.75).

Edgar, R. C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**(5):1792-1797. DOI: [doi:10.1093/nar/gkh340](https://doi.org/10.1093/nar/gkh340).

Edgar, R. C. (2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* **5**:113. DOI: [doi:10.1186/147121055113](https://doi.org/10.1186/147121055113).

See Also

[msaClustalW](#), [msaClustalOmega](#), [msaMuscle](#), [msaPrettyPrint](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## call unified interface msa() for default method (ClustalW) and
## default parameters
msa(mySeqs)

## call ClustalOmega through unified interface
```

```
msa(mySeqs, method="ClustalOmega")

## call MUSCLE through unified interface with some custom parameters
msa(mySeqs, method="Muscle", gapOpening=12, gapExtension=3, maxiters=16,
    cluster="upgmamax", SUEFF=0.4, brenner=FALSE,
    order="input", verbose=FALSE)
```

msaCheckNames	<i>Check and fix sequence names</i>
---------------	-------------------------------------

Description

This function checks and fixed sequence names of multiple alignment objects if they contain characters that might lead to LaTeX problems when using [msaPrettyPrint](#).

Usage

```
msaCheckNames(x, replacement=" ", verbose=TRUE)
```

Arguments

x	an object of class MultipleAlignment (which includes objects of classes MsaAAMultipleAlignment , MsaDNAMultipleAlignment , and MsaRNAMultipleAlignment)
replacement	a character string specifying with which character(s) potentially problematic characters should be replaced.
verbose	if TRUE (default), a warning message is shown if potentially problematic characters are found. Otherwise, the function silently replaces these characters (see details below).

Details

The **Biostrings** package does not impose any restrictions on the names of sequences. Consequently, **msa** also allows all possible ASCII strings as sequence (row) names in multiple alignments. As soon as [msaPrettyPrint](#) is used for pretty-printing multiple sequence alignments, however, the sequence names are interpreted as plain LaTeX source code. Consequently, LaTeX errors may arise because of characters or words in the sequence names that LaTeX does not or cannot interpret as plain text correctly. This particularly includes appearances of special characters and backslash characters in the sequence names.

The `msaCheckNames` function takes a multiple alignment object and checks sequence names for possibly problematic characters, which are all characters but letters (upper and lower case), digits, spaces, commas, colons, semicolons, periods, question and exclamation marks, dashes, braces, single quotes, and double quotes. All other characters are considered problematic. The function allows for both checking and fixing the sequence names. If called with `verbose=TRUE` (default), the function prints a warning if a problematic character is found. At the same time, regardless of the `verbose` argument, the function invisibly returns a copy of `x` in whose sequence names all problematic characters have been replaced by the string that is supplied via the `replacement` argument (the default is a single space).

In any case, the best solution is to check sequence names carefully and to avoid problematic sequence names from the beginning.

Value

The function invisibly returns a copy of the argument `x` (therefore, an object of the same class as `x`), but with modified sequence/row names (see details above).

Author(s)

Ulrich Bodenhofer

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

See Also

[msaPrettyPrint](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#)

Examples

```
## create toy example
mySeqs <- DNASTringSet(c("ACGATCGATC", "ACGACGATC", "ACGATCCCC"))
names(mySeqs) <- c("Seq. #1", "Seq. \2", "Seq. ~3")

## perform multiple alignment
myAlignment <- msa(mySeqs)
myAlignment

## check names
msaCheckNames(myAlignment)

## fix names
myAlignment <- msaCheckNames(myAlignment, replacement="", verbose=FALSE)
myAlignment
```

msaClustalOmega

Multiple Sequence Alignment with ClustalOmega

Description

This function calls the multiple sequence alignment algorithm ClustalOmega.

Usage

```
msaClustalOmega(inputSeqs, cluster="default",
               gapOpening="default", gapExtension="default",
               maxiters="default", substitutionMatrix="default",
               type="default", order=c("aligned", "input"),
               verbose=FALSE, help=FALSE, ...)
```

Arguments

inputSeqs	input sequences; see msa . In the original ClustalOmega implementation, this parameter is called <code>infile</code> .
cluster	The cluster size which should be used. The default is 100. In the original ClustalOmega implementation, this parameter is called <code>cluster-size</code> .
gapOpening, gapExtension	ClustalOmega currently does not allow to adjust gap penalties; these arguments are only for future extensions and consistency with the other algorithms and msa . However, setting these parameters to values other than "default" will result in a warning.
maxiters	maximum number of iterations; the default value is 0 (no limitation). In the original ClustalOmega implementation, this parameter is called <code>iterations</code> .
substitutionMatrix	name of substitution matrix for scoring matches and mismatches; can be one of the choices "BLOSUM30", "BLOSUM40", "BLOSUM50", "BLOSUM65", "BLOSUM80", and "Gonnet". This parameter is a new feature - the original ClustalOmega implementation does not allow for using a custom substitution matrix.
type	type of the input sequences <code>inputSeqs</code> ; see msa .
order	how the sequences should be ordered in the output object (see msa); in the original ClustalW implementation, this parameter is called <code>output-order</code> .
verbose	if TRUE, the algorithm displays detailed information and progress messages.
help	if TRUE, information about algorithm-specific parameters is displayed. In this case, no multiple sequence alignment is performed and the function quits after displaying the additional help information.
...	further parameters specific to ClustalOmega; An overview of parameters that are available in this interface is shown when calling <code>msaClustalOmega</code> with <code>help=TRUE</code> . For more details, see also the documentation of ClustalOmega.

Details

This is a function providing the ClustalOmega multiple alignment algorithm as an R function. It can be used for various types of sequence data (see `inputSeqs` argument above). Parameters that are common to all multiple sequences alignments provided by the **msa** package are explicitly provided by the function and named in the same for all algorithms. Most other parameters that are specific to ClustalOmega can be passed to ClustalOmega via additional arguments (see argument `help` above).

Since ClustalOmega only allows for using built-in amino acid substitution matrices, it is hardly useful for multiple alignments of nucleotide sequences.

For a note on the order of output sequences and direct reading from FASTA files, see [msa](#).

Value

Depending on the type of sequences for which it was called, `msaClustalOmega` returns a `MsaAAMultipleAlignment`, `MsaDNAMultipleAlignment`, or `MsaRNAMultipleAlignment` object. If called with `help=TRUE`, `msaClustalOmega` returns an invisible `NULL`.

Author(s)

Enrico Bonatesta and Christoph Kainrath

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). `msa`: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

<http://www.clustal.org/omega/README>

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Soeding, J., Thompson, J. D., and Higgins, D. G. (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.* **7**:539. DOI: [doi:10.1038/msb.2011.75](https://doi.org/10.1038/msb.2011.75).

See Also

[msa](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## call msaClustalOmega with default values
msaClustalOmega(mySeqs)

## call msaClustalOmega with custom parameters
msaClustalOmega(mySeqs, auto=FALSE, cluster=120, dealign=FALSE,
                useKimura=FALSE, order="input", verbose=FALSE)
```

msaClustalW

Multiple Sequence Alignment with ClustalW

Description

This function calls the multiple sequence alignment algorithm `ClustalW`.

Usage

```
msaClustalW(inputSeqs, cluster="default", gapOpening="default",
            gapExtension="default", maxiters="default",
            substitutionMatrix="default", type="default",
            order=c("aligned", "input"), verbose=FALSE,
            help=FALSE, ...)
```

Arguments

inputSeqs	input sequences; see msa . In the original ClustalW implementation, this parameter is called <code>infile</code> .
cluster	The clustering method which should be used. Possible values are "nj" (default) and "upgma". In the original ClustalW implementation, this parameter is called <code>clustering</code> . Please note that <code>cluster="upgma"</code> leads to an unidentified error on Windows with R 4.0.x that even crashes the entire R session.
gapOpening	gap opening penalty; the default value for nucleotide sequences is 15.0, the default value for amino acid sequences is 10.0.
gapExtension	gap extension penalty; the default value for nucleotide sequences is 6.66, the default value for amino acid sequences is 0.2.
maxiters	maximum number of iterations; the default value is 16. In the original ClustalW implementation, this parameter is called <code>numiters</code> .
substitutionMatrix	substitution matrix for scoring matches and mismatches; can be a real matrix, a file name, or the name of a built-in substitution matrix. In the latter case, the choices "blosum", "pam", "gonnet", and "id" are supported for amino acid sequences. For aligning nucleotide sequences, the choices "iub" and "clustalw" are possible. The parameter <code>dnamatrix</code> can also be used instead for the sake of backwards compatibility. The valid choices for this parameter are "iub" and "clustalw". In the original ClustalW implementation, this parameter is called <code>matrix</code> .
type	type of the input sequences <code>inputSeqs</code> ; see msa .
order	how the sequences should be ordered in the output object (see msa); in the original ClustalW implementation, this parameter is called <code>outorder</code> .
verbose	if TRUE, the algorithm displays detailed information and progress messages.
help	if TRUE, information about algorithm-specific parameters is displayed. In this case, no multiple sequence alignment is performed and the function quits after displaying the additional help information.
...	further parameters specific to ClustalW; An overview of parameters that are available in this interface is shown when calling <code>msaClustalW</code> with <code>help=TRUE</code> . For more details, see also the documentation of ClustalW.

Details

This is a function providing the ClustalW multiple alignment algorithm as an R function. It can be used for various types of sequence data (see `inputSeqs` argument above). Parameters that are

common to all multiple sequences alignments provided by the **msa** package are explicitly provided by the function and named in the same for all algorithms. Most other parameters that are specific to ClustalW can be passed to ClustalW via additional arguments (see argument help above).

For a note on the order of output sequences and direct reading from FASTA files, see [msa](#).

Value

Depending on the type of sequences for which it was called, `msaClustalW` returns a [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), or [MsaRNAMultipleAlignment](#) object. If called with `help=TRUE`, `msaClustalW` returns an invisible `NULL`.

Author(s)

Enrico Bonatesta and Christoph Kainrath

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). `msa`: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

http://www.clustal.org/download/clustalw_help.txt

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**(22):4673-4680. DOI: [doi:10.1093/nar/22.22.4673](https://doi.org/10.1093/nar/22.22.4673).

See Also

[msa](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## call msaClustalW with default values
msaClustalW(mySeqs)

## call msaClustalW with custom parameters
msaClustalW(mySeqs, gapOpening=1, gapExtension=1, maxiters=16,
             kimura=FALSE, order="input", maxdiv=23)
```

 msaConsensusSequence *Computation of Consensus Sequence from Multiple Alignment*

Description

This method computes a consensus sequence from a multiple alignment or a previously computed consensus matrix. Currently, two different ways of these computations are available.

Usage

```
## S4 method for signature 'matrix'
msaConsensusSequence(x, type=c("Biostrings", "upperlower"),
  thresh=c(80, 20), ignoreGaps=FALSE, ...)
## S4 method for signature 'MultipleAlignment'
msaConsensusSequence(x, ...)
```

Arguments

x	an object of class MultipleAlignment (which includes objects of classes MsaAAMultipleAlignment , MsaDNAMultipleAlignment , and MsaRNAMultipleAlignment) or a previously computed consensus matrix (see details below).
type	a character string specifying how to compute the consensus sequence. Currently, types "Biostrings" and "upperlower" are available (see details below).
thresh	a decreasing two-element numeric vector of numbers between 0 and 100 corresponding to the two conservation thresholds. Only relevant for type="upperlower" (see details below), otherwise ignored.
ignoreGaps	a logical (default: FALSE) indicating whether gaps should be considered when computing the consensus sequence. Only relevant for type="upperlower" (see details below), otherwise ignored.
...	when the method is called for a MultipleAlignment object, the consensus matrix is computed and, including all further arguments, passed on to the <code>msaConsensusSequence</code> method with signature <code>matrix</code> . The method with signature <code>matrix</code> forwards additional arguments to the consensusString method from the Biostrings package if type="Biostrings".

Details

The method takes a [MultipleAlignment](#) object or a previously computed consensus matrix and computes a consensus sequence. For type="Biostrings", the method [consensusString](#) from the **Biostrings** package is used to compute the consensus sequence. For type="upperlower", two thresholds (argument `thresh`, see above) are used to compute the consensus sequence:

- If the relative frequency of the most frequent letter at a given position is at least as large as the first threshold (default: 80%), then this most frequent letter is used for the consensus sequence at this position as it is.

- If the relative frequency of the most frequent letter at a given position is smaller than the first threshold, but at least as large as the second threshold (default: 20%), then this most frequent letter is used for the consensus sequence at this position, but converted to lower case.
- If the relative frequency of the most frequent letter in a column is even smaller than the second threshold, then a dot is used for the consensus sequence at this position.
- If ignoreGaps=FALSE (which is the default), gaps are treated like all other letters except for the fact that obviously no lowercase conversion takes place in case that the relative frequency is between the two thresholds. If ignoreGaps=TRUE, gaps are ignored completely, and the consensus sequence is computed from the non-gap letters only.

If the consensus matrix of a multiple alignment of nucleotide sequences contains rows labeled '+' and/or '.', these rows are ignored.

Value

The function returns a character string with the consensus sequence.

Author(s)

Ulrich Bodenhofer

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

See Also

[msa](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#), [MultipleAlignment](#), [consensusString](#)

Examples

```
## read sequences
filepath <- system.file("examples", "HemoglobinAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## perform multiple alignment
myAlignment <- msa(mySeqs)

## regular consensus sequence using consensusString() method from the
## 'Biostrings' package
msaConsensusSequence(myAlignment)

## use the other method
msaConsensusSequence(myAlignment, type="upperlower")

## use the other method with custom parameters
```

```

msaConsensusSequence(myAlignment, type="upperlower", thresh=c(50, 20),
                      ignoreGaps=TRUE)

## compute a consensus matrix first
conMat <- consensusMatrix(myAlignment)
msaConsensusSequence(conMat)

```

msaConservationScore *Computation of Conservation Scores from Multiple Alignment*

Description

This method computes a vector of conservation scores from a multiple alignment or a previously computed consensus matrix.

Usage

```

## S4 method for signature 'matrix'
msaConservationScore(x, substitutionMatrix, gapVsGap=NULL,
                    ...)
## S4 method for signature 'MultipleAlignment'
msaConservationScore(x, ...)

```

Arguments

x	an object of class <code>MultipleAlignment</code> (which includes objects of classes <code>MsaAAMultipleAlignment</code> , <code>MsaDNAMultipleAlignment</code> , and <code>MsaRNAMultipleAlignment</code>) or a previously computed consensus matrix (see details below).
substitutionMatrix	substitution matrix (see details below).
gapVsGap	score to use for aligning gaps versus gaps (see details below).
...	when the method is called for a <code>MultipleAlignment</code> object, the consensus matrix is computed and, including all further arguments, passed on to the <code>msaConservationScore</code> method with signature <code>matrix</code> . This method passes all further arguments on to the <code>msaConsensusSequence</code> method to customize the way the consensus sequence is computed.

Details

The method takes a `MultipleAlignment` object or a previously computed consensus matrix and computes the sum of pairwise scores for all positions of the alignment. For computing these scores, it is compulsory to specify a substitution/scoring matrix. This matrix must be provided as a `matrix` object. This can either be one of the ready-made matrices provided by the `pwalign` package (e.g. `BLOSUM62`) or any other hand-crafted matrix. In the latter case, the following restrictions apply:

- The matrix must be quadratic.
- For reasonable results, the matrix should be symmetric (note that this is not checked).

- Rows and columns must be named and the order of letters/symbols in row names and column names must be identical.
- All letters/symbols occurring in the multiple alignment, including gaps '-', must also be found in the row/column names of the substitution matrix. For consistency with the matrices from the **pwalign** package, the row/column corresponding to gap penalties may also be labeled '*' instead of '-'.

So, nucleotide substitution matrices created by `nucleotideSubstitutionMatrix` can be used for multiple alignments of nucleotide sequences, but must be completed with gap penalty rows and columns (see example below).

If the consensus matrix of a multiple alignment of nucleotide sequences contains rows labeled '+' and/or '.', these rows are ignored.

The parameter `gapVsGap` can be used to control how pairs of gaps are scored. If `gapVsGap=NULL` (default), the corresponding diagonal entry of the substitution matrix is used as is. In the BLO-SUM matrices, this is usually a positive value, which may not make sense under all circumstances. Therefore, the parameter `gapVsGap` can be set to an alternative value, e.g. 0 for ignoring gap-gap pairs.

The method, in any case, returns a vector of scores that is as long as the alignment/consensus matrix has columns. The names of the vector entries are the corresponding positions of the consensus sequence of the alignment. How this consensus sequence is computed, can be controlled with additional arguments that are passed on to the `msaConsensusSequence` method.

Value

The function returns a vector as long as the alignment/consensus matrix has columns. The vector is named with the consensus sequence (see details above).

Author(s)

Ulrich Bodenhofer

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatista, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

See Also

[msa](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#), [MultipleAlignment](#), [msaConsensusSequence](#)

Examples

```
## load 'pwalign' package which provides substitution matrices
library(pwalign)

## read sequences
```



```

filepath <- system.file("examples", "HemoglobinAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## perform multiple alignment
myAlignment <- msa(mySeqs)

## compute consensus scores using the BLOSUM62 matrix
data(BLOSUM62)
msaConservationScore(myAlignment, BLOSUM62)

## compute consensus scores using the BLOSUM62 matrix
## without scoring gap-gap pairs and using a different consensus sequence
msaConservationScore(myAlignment, BLOSUM62, gapVsGap=0,
                    type="upperlower")

## compute a consensus matrix first
conMat <- consensusMatrix(myAlignment)
data(PAM250)
msaConservationScore(conMat, PAM250, gapVsGap=0)

## DNA example
filepath <- system.file("examples", "exampleDNA.fasta", package="msa")
mySeqs <- readDNAStrngSet(filepath)

## perform multiple alignment
myAlignment <- msa(mySeqs)

## create substitution matrix with gap penalty -8
mat <- nucleotideSubstitutionMatrix(4, -1)
mat <- cbind(rbind(mat, "-"=-8), "-"=-8)

## compute consensus scores using this matrix
msaConservationScore(myAlignment, mat, gapVsGap=0)

```

msaConvert

Convert Multiple Sequence Alignment for Other Packages

Description

This function converts a multiple sequence alignment object to formats used in other sequence analysis packages.

Usage

```

msaConvert(x,
          type=c("seqinr::alignment", "bios2mds::align",
                "ape::AAbin", "ape::DNAbin",
                "phangorn::phyDat", "bio3d::fasta"))

```

Arguments

x	an object of class <code>MultipleAlignment</code> (which includes objects of classes <code>MsaAAMultipleAlignment</code> , <code>MsaDNAMultipleAlignment</code> , and <code>MsaRNAMultipleAlignment</code>)
type	a character string specifying to which type of object x should be converted; currently, the values <code>"seqinr::alignment"</code> , <code>"bios2mds::align"</code> , <code>"ape::AAbin"</code> , <code>"ape::DNAbin"</code> , <code>"phangorn::phyDat"</code> , and <code>"bio3d::fasta"</code> .

Details

The function converts x to the class of object as specified by the type argument. The values possible for the type argument follow the same principle `pkg::cl`, i.e. x is converted to class `cl` as defined in the `pkg` package.

The conversions for usage by the packages **seqinr**, **bios2mds**, and **ape** work independently of these packages and do not strictly require these packages. They need not even be installed. This approach has been chosen to avoid abundant dependencies and possible incompatibilities. That is also why the standard S3/S4 mechanism of `as/as.class` functions is not used.

The conversion to the `phyDat` class can be done easily using the `as.phyDat` function from the **phangorn** package. The `msaConvert` function still provides this conversion for the sake of consistency. However, this conversion is just a wrapper function around the `as.phyDat` function from the **phangorn** package. Thus, the **phangorn** package needs to be installed.

The conversion `"ape::AAbin"` only works for multiple alignments of amino acid sequences, while the conversions `"ape::DNAbin"` and `"phangorn::phyDat"` only work for multiple alignments of DNA sequences. When converting to `"ape::AAbin"`, gaps/dashes are replaced by 'X'. Moreover, conversions to `"ape::DNAbin"` also convert all characters to lowercase and replace gaps/dashes by 'n'.

Value

The function returns an object of the class as specified by the type argument.

Author(s)

Ulrich Bodenhofer

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

See Also

[msa](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## perform multiple alignment
myAlignment <- msa(mySeqs)

## convert to an object of class 'alignment' (package 'seqinr')
msaConvert(myAlignment, "seqinr::alignment")

## convert to an object of class 'align' (package 'bios2mds')
msaConvert(myAlignment, "bios2mds::align")
```

MsaMetaData-class	Class MsaMetaData
-------------------	-------------------

Description

S4 class for storing metadata about multiple sequence alignment results

Objects

Objects of this virtual class are not be created and used directly. This is an auxiliary class used by the classes [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), and [MsaRNAMultipleAlignment](#)

Slots

The following slots are defined for MsaMetaData objects:

version: slot in which information is stored with which algorithm the multiple alignment has been computed along with its version number.

params: list in which the parameters are stored with which the multiple alignment algorithm has been executed.

call: the matched call with which the object was created

Methods

version(object): accessor to the version slot

params(x): accessor to the params slot

Author(s)

Enrico Bonatesta and Christoph Kainrath

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

See Also

[msa](#), [msaClustalW](#), [msaClustalOmega](#), [msaMuscle](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## simple call with default values
myAlignment <- msaClustalOmega(mySeqs)

## show the algorithm version with which the results were created
version(myAlignment)

## show the results
show(myAlignment)

## print the results
print(myAlignment, show="alignment")
print(myAlignment, show=c("alignment", "version"))
print(myAlignment, show="standardParams")
print(myAlignment, show="algParams")
print(myAlignment, show=c("call", "version"))

## show the params
params(myAlignment)
```

MsaMultipleAnlignmentClasses

Classes MsaAAMultipleAlignment, MsaDNAMultipleAlignment,
and MsaRNAMultipleAlignment

Description

S4 classes for storing multiple alignments of amino acid, DNA, and RNA sequences along with algorithm metadata

Objects

Objects of these classes are returned by the multiple sequence alignment algorithms [msaClustalW](#), [msaClustalOmega](#), [msaMuscle](#), and the wrapper function [msa](#), all of which are provided by the **msa** package.

Details

The class `MsaAAMultipleAlignment` extends the `AAMultipleAlignment` class, the class `MsaDNAMultipleAlignment` extends the `DNAMultipleAlignment` class, and the class `MsaRNAMultipleAlignment` extends the `RNAMultipleAlignment` class. All three classes extend their parent classes by the slots contained in the `MsaMetaData`, i.e. all three classes are class unions of the aforementioned parent classes and the class `MsaMetaData`.

Methods

`print(x, show=c("alignment", "version", "call"), showNames=TRUE, showConsensus=TRUE, halfNrow=9, nameWidth)` prints information about the object `x`; the `show` argument allows for determining what should be printed. The `show` must be a character vector and may contain any combination of the following strings: if `show` contains "alignment", the multiple sequence alignment is printed in a way similar to the corresponding method from the **Biostrings** package (except for the consensus sequence, see below). If `show` contains "complete", the entire width of the alignment is printed by splitting it over multiple blocks of lines if necessary. This overrules "alignment" if both are contained in the `show` argument. If `show` contains "version", the version slot is shown. If `show` contains "call", the call slot is shown. If `show` contains "standardParams", the settings of the parameters that are common to all three multiple sequence alignment algorithms are shown. If `show` contains "algParams", the algorithm-specific parameters are shown. The order in which the strings are placed in the `show` argument does not have an effect on the order in which data are printed. The default is `show=c("alignment", "version", "call")`, i.e. by default, the multiple sequence alignment is shown along with version and call information. If `show` contains "all", the complete alignment is shown along with version information, call, and the complete set of parameters. As said above, by default, printing alignments is similar to the standard `print` method provided by the **Biostrings** package, whereas including "complete" in the argument `show` prints the entire width of the alignment. Unlike the method from the **Biostrings** package, the appearance can be customized: by default, the consensus sequence is appended below the alignment. To switch this off, use `showConsensus=FALSE`. Whether or not sequence names should be printed can be controlled via the `showNames` argument. The width reserved for the sequence names can be adjusted using the `nameWidth` argument; the default is 20 like in the **Biostrings** method. If the number of sequences in the alignment is large, output can become quite lengthy. That is why only the first `halfNrow` and the last `halfNrow` sequences are shown. To show all sequences, set `halfNrow` to NA or -1. Note that `print` can also handle masked objects, where the masked sequences/positions are shown as hash marks. However, the consensus sequences are computed from the complete, unmasked alignment and displayed as such. Additional arguments are passed on to [msaConsensusSequence](#) for customizing how the consensus sequence is computed.

`show(object)`: displays the alignment along with metadata; synonymous to calling `print` with default arguments.

`version(object)`: displays the algorithm with which the multiple alignment has been computed along with its version number (see also [MsaMetaData](#)).

`params(x)`: accessor to the `params` slot (see also [MsaMetaData](#))

Author(s)

Enrico Bonatesta, Christoph Kainrath, and Ulrich Bodenhofer

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). `msa`: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

See Also

[msa](#), [msaClustalW](#), [msaClustalOmega](#), [msaMuscle](#), [MsaMetaData](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## simple call with default values
myAlignment <- msaClustalOmega(mySeqs)

## show the algorithm version with which the results were created
version(myAlignment)

## show the results
show(myAlignment)

## print the results
print(myAlignment, show="alignment")
print(myAlignment, show="alignment", showConsensus=FALSE)
print(myAlignment, show=c("alignment", "version"))
print(myAlignment, show="standardParams")
print(myAlignment, show="algParams")
print(myAlignment, show=c("call", "version"))

## print results with custom consensus sequence
print(myAlignment, show="complete", type="upperlower", thresh=c(50, 20))

## show the params
params(myAlignment)
```

msaMuscle

*Multiple Sequence Alignment with MUSCLE***Description**

This function calls the multiple sequence alignment algorithm MUSCLE.

Usage

```
msaMuscle(inputSeqs, cluster="default", gapOpening="default",
          gapExtension="default", maxiters="default",
          substitutionMatrix="default",
          type="default", order=c("aligned", "input"),
          verbose=FALSE, help=FALSE, ...)
```

Arguments

inputSeqs	input sequences; see msa . In the original MUSCLE implementation, this parameter is called <code>-in</code> .
cluster	The clustering method which should be used. Possible values are "upgma", "upgmamax", "upgmamin", "upgmb", and "neighborjoining".
gapOpening	gap opening penalty; the default is 400 for DNA sequences and 420 for RNA sequences. The default for amino acid sequences depends on the profile score settings: for the setting <code>le=TRUE</code> , the default is 2.9, for <code>sp=TRUE</code> , the default is 1,439, and for <code>sv=TRUE</code> , the default is 300. Note that these defaults may not be suitable if custom substitution matrices are being used. In such a case, a sensible choice of gap penalties that fits well to the substitution matrix must be made.
gapExtension	gap extension penalty; the default is 0.
maxiters	maximum number of iterations; the default is 16. In the original MUSCLE implementation, it is also possible to set <code>maxiters</code> to 0 which leads to an (out of memory) error. Therefore, <code>maxiters=0</code> is not allowed in <code>msaMuscle</code> .
substitutionMatrix	substitution matrix for scoring matches and mismatches; can be a real matrix or a file name. If the file interface is used, matrices have to be in NCBI-format. The original MUSCLE implementation also accepts matrices in WU_BLAST (AB_BLAST) format, but, due to copyright restrictions, this format is not supported by <code>msaMuscle</code> .
type	type of the input sequences <code>inputSeqs</code> ; see msa .
order	how the sequences should be ordered in the output object (see msa for more details); the original MUSCLE implementation does not allow for preserving the order of input sequences. The <code>msaMuscle</code> function realizes this functionality by reverse matching of sequence names. Therefore, the sequences need to have unique names. If the sequences do not have names or if the names are not unique, the <code>msaMuscle</code> function assigns generic unique names "Seq1"-Seqn to the sequences and issues a warning. The choice "input" is not available at all for sequence data that is read directly from a FASTA file.

verbose	if TRUE, the algorithm displays detailed information and progress messages.
help	if TRUE, information about algorithm-specific parameters is displayed. In this case, no multiple sequence alignment is performed and the function quits after displaying the additional help information.
...	further parameters specific to MUSCLE; An overview of parameters that are available in this interface is shown when calling <code>msaMuscle</code> with <code>help=TRUE</code> . For more details, see also the documentation of MUSCLE.

Details

This is a function providing the MUSCLE multiple alignment algorithm as an R function. It can be used for various types of sequence data (see `inputSeqs` argument above). Parameters that are common to all multiple sequences alignments provided by the **msa** package are explicitly provided by the function and named in the same for all algorithms. Most other parameters that are specific to MUSCLE can be passed to MUSCLE via additional arguments (see argument `help` above).

For a note on the order of output sequences and direct reading from FASTA files, see [msa](#).

Value

Depending on the type of sequences for which it was called, `msaMuscle` returns a [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), or [MsaRNAMultipleAlignment](#) object. If called with `help=TRUE`, `msaMuscle` returns an invisible NULL.

Author(s)

Enrico Bonatesta and Christoph Kainrath

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

<http://www.drive5.com/muscle/muscle.html>

Edgar, R. C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**(5):1792-1797. DOI: [doi:10.1093/nar/gkh340](https://doi.org/10.1093/nar/gkh340).

Edgar, R. C. (2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* **5**:113. DOI: [doi:10.1186/147121055113](https://doi.org/10.1186/147121055113).

See Also

[msa](#), [MsaAAMultipleAlignment](#), [MsaDNAMultipleAlignment](#), [MsaRNAMultipleAlignment](#), [MsaMetaData](#)

Examples

```
## load 'pwalign' package which provides substitution matrices
library(pwalign)

## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## call msaMuscle with default values
msaMuscle(mySeqs)

## call msaMuscle with custom parameters
msaMuscle(mySeqs, gapOpening=12, gapExtension=3, maxiters=16,
          cluster="upgmamax", SUEFF=0.4, brenner=FALSE,
          order="input", verbose=FALSE)

## call msaMuscle with a custom substitution matrix
data(PAM120)
msaMuscle(mySeqs, substitutionMatrix=PAM120)
```

msaPrettyPrint

Pretty-Printing of Multiple Sequence Alignments

Description

The `msaPrettyPrint` function provides an R interface to the powerful LaTeX package `texshade.sty` which allows for a highly customizable plots of multiple sequence alignments.

Usage

```
msaPrettyPrint(x, y, output=c("pdf", "tex", "dvi", "asis"),
              subset=NULL, file=NULL, alFile=NULL,
              askForOverwrite=TRUE, psFonts=FALSE, code=NA,
              paperWidth=11, paperHeight=8.5, margins=c(0.1, 0.3),
              shadingMode=c("identical", "similar", "functional"),
              shadingModeArg=NA,
              shadingColors=c("blues", "reds", "greens", "grays",
                              "black"),
              showConsensus=c("bottom", "top", "none"),
              consensusColors=c("ColdHot", "HotCold", "BlueRed",
                                 "RedBlue", "GreenRed",
                                 "RedGreen", "Gray"),
              consensusThreshold=50,
              showLogo=c("top", "bottom", "none"),
              logoColors=c("chemical", "rasmol", "hydropathy",
                           "structure", "standard area",
                           "accessible area"),
              showLogoScale=c("none", "leftright", "left"),
```

```

"right"),
showNames=c("left", "right", "none"),
showNumbering=c("right", "left", "none"),
showLegend=TRUE, furtherCode=NA, verbose=FALSE)

```

Arguments

x	an object of class <code>MultipleAlignment</code> , which includes the classes <code>MsaAAMultipleAlignment</code> , <code>MsaDNAMultipleAlignment</code> , and <code>MsaRNAMultipleAlignment</code> .
y	argument for restricting the output to a subset of columns; can be a numeric vector of length 2 with a lower and an upper bound or an object of class <code>IRanges</code> . If missing, the entire multiple alignment is printed.
output	type of output to be generated (see details below)
subset	can be used to specify a subset of sequences in the multiple alignment x if not all sequences should be printed.
file	name of output file; if no name is given, the name of the output file defaults to name of the object provided as argument x along with the proper suffix which depends on the type of output specified with the output argument. Note that this might lead to invalid file names if not the name of an object, but an R expression is passed as argument x.
alFile	name of alignment file to be created; <code>msaPrettyPrint</code> first writes the multiple alignment x to a <code>.fasta</code> file. The name of this file can be determined with the <code>alFile</code> argument. If no name is given, the name of the output file defaults to name of the object provided as argument x along with the suffix <code>.fasta</code> . Note that this might lead to invalid file names if not the name of an object, but an R expression is passed as argument x.
askForOverwrite	if TRUE (default), <code>msaPrettyPrint</code> asks whether existing files should be overwritten or not. If <code>askForOverwrite</code> is set to FALSE, files are overwritten without further notice.
psFonts	if TRUE, <code>msaPrettyPrint</code> produces LaTeX code that includes the LaTeX package <code>times.sty</code> ; if FALSE, <code>msaPrettyPrint</code> produces LaTeX code based on the standard LaTeX fonts (default). Ignored for <code>output="asis"</code> .
code	this argument can be used to specify the entire LaTeX code in the <code>texshade</code> environment. This overrides all arguments that customize the appearance of the output. Instead, all customizations must be done as LaTeX commands provided by the package <code>texshade.sty</code> directly. This option should only be used by expert users and for special applications in which the possibilities of the customizations of the <code>msaPrettyPrint</code> function turn out to be insufficient.
paperWidth, paperHeight	paper format to be used in the resulting document; defaults to 11in x 8.5in (US letter in landscape orientation). Ignored for <code>output="asis"</code> .
margins	a numeric vector of length 2 with the horizontal and vertical margins, respectively; the default is 0.1in for the horizontal and 0.3in for the vertical margin.
shadingMode	shading mode; currently the shading modes <code>"identical"</code> , <code>"similar"</code> , and <code>"functional"</code> are supported (see documentation of <code>texshade.sty</code> for details).

shadingModeArg	for shading modes "identical" and "similar", shadingModeArg must be a single numeric threshold between 0 and 100 or two thresholds between 0 and 100 in increasing order. For shading mode "functional", valid shadingModeArg arguments are "charge", "hydropathy", "structure", "chemical", "rasmol", "standard area", and "accessible area" (see documentation of texshade.sty for details).
shadingColors	color scheme for shading; valid "shadingColors" arguments are "blues", "reds", "greens", "grays", and "black" (see documentation of texshade.sty for details).
showConsensus	where to show the consensus sequence; possible values are "bottom", "top", and "none" (the latter option suppresses printing of the consensus sequence).
consensusColors	color scheme for printing the consensus sequence; the following choices are possible: "ColdHot", "HotCold", "BlueRed", "RedBlue", "GreenRed", "RedGreen", and "Gray" (see documentation of texshade.sty for details).
consensusThreshold	one or two numbers between 0 and 100, where the second one is optional and must be larger than the first one (see documentation of texshade.sty for details)
showLogo	where to show a sequence logo; possible values are "top", "bottom", or "none" (the latter option suppresses printing of the consensus sequence). If a sequence logo and a consensus sequence should be shown together, they can only be located at opposite sides.
logoColors	color scheme for printing the sequence logo; the following choices are possible: "chemical", "rasmol", "hydropathy", "structure", "standard area", and "accessible area" (see documentation of texshade.sty for details).
showLogoScale	where to plot the vertical axis of the sequence logo; possible values are "left", "right", "leftright", and "none" (the latter option suppresses that the axis is displayed).
showNames	where to print sequence names; possible values are "left", "right", and "none" (the latter option suppresses that names are displayed).
showNumbering	where to print sequence numbers; possible values are "left", "right", and "none" (the latter option suppresses that numbers are displayed). If sequence names and numbers should be shown together, they can only be located at opposite sides.
showLegend	if TRUE (default), a legend is printed at the end of the alignment.
furtherCode	additional LaTeX code to be included in the texshade environment; all text passed as furtherCode is placed between the commands created by msaPrettyPrint and the end of the texshade environment. Note the difference to the code argument: while the code argument replaces all LaTeX code in the texshade environment, the code passed as furtherCode argument is added to the LaTeX code in the texshade environment.
verbose	if TRUE (default), progress messages are printed and also the output of running (PDF)LaTeX (if applicable) is printed to the R session.

Details

The `msaPrettyPrint` function writes a multiple alignment to a `.fasta` file and creates LaTeX code for pretty-printing the multiple alignment on the basis of the LaTeX package `texshade.sty`. If `output="asis"`, `msaPrettyPrint` prints a LaTeX fragment consisting of the `texshade` environment to the console. The parameters described above can be used to customize the way the multiple alignment is formatted. If `output="tex"`, a complete LaTeX file including preamble is created. For `output="dvi"` and `output="pdf"`, the same kind of LaTeX file is created, but processed using (PDF)LaTeX to produce a final DVI or PDF file, respectively. The `file` argument be used to determine the file name of the final output file (except for the `output="asis"` which does not create an output file).

The choice `output="asis"` is particularly useful for Sweave or knitr documents. If `msaPrettyPrint` is called with `output="asis"` in a code chunk with `results="tex"` (Sweave) or `results="asis"` (knitr), then the resulting LaTeX fragment consisting of the `texshade` environment is directly included in the LaTeX document that is created from the Sweave/knitr document.

As noted above, if they are not specified explicitly, output file names are determined automatically. It is important to point out that all file names need to be LaTeX-compliant, i.e. no special characters and spaces (!) are allowed. If a file name would be invalid, `msaPrettyPrint` makes a default choice.

Moreover, if sequence names are to be printed, there might be names that are not LaTeX-compliant and lead to LaTeX errors. In order to check that in advance, the function `msaCheckNames` is available.

Note that `texi2dvi` and `texi2pdf` always save the resulting DVI/PDF files to the current working directory, even if the LaTeX source file is in a different directory. That is also the reason why the temporary file is created in the current working directory in the example below.

Value

`msaPrettyPrint` returns an invisible character vector consisting of the LaTeX fragment with the `texshade` environment.

Author(s)

Ulrich Bodenhofer, Enrico Bonatesta, and Christoph Kainrath

References

<https://github.com/UBod/msa>

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., and Hochreiter, S. (2015). `msa`: an R package for multiple sequence alignment. *Bioinformatics* **31**(24):3997-3999. DOI: [doi:10.1093/bioinformatics/btv494](https://doi.org/10.1093/bioinformatics/btv494).

<https://www.ctan.org/pkg/texshade>

Beitz, E. (2000). TeXshade: shading and labeling of multiple sequence alignments using LaTeX2e. *Bioinformatics* **16**(2):135-139. DOI: [doi:10.1093/bioinformatics/16.2.135](https://doi.org/10.1093/bioinformatics/16.2.135).

See Also

[msaCheckNames](#)

Examples

```
## read sequences
filepath <- system.file("examples", "exampleAA.fasta", package="msa")
mySeqs <- readAAStringSet(filepath)

## call unified interface msa() for default method (ClustalW) and
## default parameters
myAlignment <- msa(mySeqs)

## show resulting LaTeX code with default settings
msaPrettyPrint(myAlignment, output="asis", askForOverwrite=FALSE)

## create PDF file according to some custom settings
## Not run:
tmpFile <- tempfile(pattern="msa", tmpdir=".", fileext=".pdf")
tmpFile
msaPrettyPrint(myAlignment, file=tmpFile, output="pdf",
               showNames="left", showNumbering="none", showLogo="top",
               showConsensus="bottom", logoColors="rasmol",
               verbose=FALSE, askForOverwrite=FALSE)

library(Biobase)
openPDF(tmpFile)
## End(Not run)
```

Index

- * **class**
 - MsaMetaData-class, 19
 - MsaMultipleAnlignmentClasses, 20
- * **graphs**
 - msaPrettyPrint, 25
- * **manip**
 - msa, 4
 - msaCheckNames, 7
 - msaClustalOmega, 8
 - msaClustalW, 10
 - msaConsensusSequence, 13
 - msaConservationScore, 15
 - msaConvert, 17
 - msaMuscle, 23
- * **package**
 - msa-package, 2
- AAMultipleAlignment, 21
- AAStringSet, 4, 5
- BLOSUM62, 15
- class:MsaAAMultipleAlignment
 - (MsaMultipleAnlignmentClasses), 20
- class:MsaDNAMultipleAlignment
 - (MsaMultipleAnlignmentClasses), 20
- class:MsaMetaData (MsaMetaData-class), 19
- class:MsaRNAMultipleAlignment
 - (MsaMultipleAnlignmentClasses), 20
- consensusString, 13, 14
- DNAMultipleAlignment, 21
- DNAStrngSet, 4, 5
- IRanges, 26
- matrix, 15
- msa, 2, 3, 4, 9–12, 14, 16, 18, 20–24
- msa-package, 2
- MsaAAMultipleAlignment, 6–8, 10, 12–16, 18–20, 24, 26
- MsaAAMultipleAlignment
 - (MsaMultipleAnlignmentClasses), 20
- MsaAAMultipleAlignment-class
 - (MsaMultipleAnlignmentClasses), 20
- msaCheckNames, 7, 28
- msaClustalOmega, 2–6, 8, 20–22
- msaClustalW, 2–6, 10, 20–22
- msaConsensusSequence, 13, 15, 16, 21
- msaConsensusSequence, matrix-method
 - (msaConsensusSequence), 13
- msaConsensusSequence, MultipleAlignment-method
 - (msaConsensusSequence), 13
- msaConservationScore, 15
- msaConservationScore, matrix-method
 - (msaConservationScore), 15
- msaConservationScore, MultipleAlignment-method
 - (msaConservationScore), 15
- msaConvert, 17
- MsaDNAMultipleAlignment, 6–8, 10, 12–16, 18–20, 24, 26
- MsaDNAMultipleAlignment
 - (MsaMultipleAnlignmentClasses), 20
- MsaDNAMultipleAlignment-class
 - (MsaMultipleAnlignmentClasses), 20
- MsaMetaData, 6, 10, 12, 14, 16, 18, 21, 22, 24
- MsaMetaData (MsaMetaData-class), 19
- MsaMetaData-class, 19
- MsaMultipleAnlignmentClasses, 20
- msaMuscle, 2–6, 20–23, 23
- msaPackage (msa-package), 2
- msaPrettyPrint, 3, 6–8, 25

- MsaRNAMultipleAlignment, [6–8](#), [10](#), [12–16](#),
[18–20](#), [24](#), [26](#)
- MsaRNAMultipleAlignment
(MsaMultipleAnlignmentClasses),
[20](#)
- MsaRNAMultipleAlignment-class
(MsaMultipleAnlignmentClasses),
[20](#)
- MultipleAlignment, [7](#), [13–16](#), [18](#), [26](#)
- nucleotideSubstitutionMatrix, [16](#)

- params, MsaAAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- params, MsaDNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- params, MsaMetaData-method
(MsaMetaData-class), [19](#)
- params, MsaRNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- print, MsaAAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- print, MsaDNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- print, MsaRNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)

- RNAMultipleAlignment, [21](#)
- RNAStringSet, [4](#), [5](#)

- show, MsaAAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- show, MsaDNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- show, MsaRNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)

- texi2dvi, [28](#)
- texi2pdf, [28](#)

- version (MsaMetaData-class), [19](#)
- version, MsaAAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- version, MsaDNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)
- version, MsaMetaData-method
(MsaMetaData-class), [19](#)
- version, MsaRNAMultipleAlignment-method
(MsaMultipleAnlignmentClasses),
[20](#)

- XStringSet, [2](#), [4](#)