Software Manual

Institute of Bioinformatics, Johannes Kepler University Linz

msa

An R Package for Multiple Sequence Alignment

Enrico Bonatesta, Christoph Horejs-Kainrath, and Ulrich Bodenhofer

Institute of Bioinformatics, Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
msa@bioinf.jku.at

Version 1.0.2, September 29, 2015

Scope and Purpose of this Document

This document provides a gentle introduction into the R package msa. Not all features of the R package are described in full detail. Such details can be obtained from the documentation enclosed in the R package. Further note the following: (1) this is not an introduction to multiple sequence alignment or algorithms for multiple sequence alignment; (2) this is not an introduction to R or any of the Bioconductor packages used in this document. If you lack the background for understanding this manual, you first have to read introductory literature on the subjects mentioned above.

Contents

1 Introduction

Multiple sequence alignment is one of the most fundamental tasks in bioinformatics. Algorithms like ClustalW [19, 9], ClustalOmega [17], and MUSCLE [5, 6] are well known and widely used (for more comprehensive overviews of methods, see [7, 14, 20]). However, all these algorithms are implemented as stand-alone commmand line programs without any integration into the R/Bioconductor ecosystem. Before the `msa` package, only the `muscle` package has been available in `R`, but no other multiple sequence alignment algorithm, although the `Biostrings` package has provided data types for representing multiple sequence alignments for quite some time [16]. The `msa` package aims to close that gap by providing a unified R interface to the multiple sequence alignment algorithms ClustalW, ClustalOmega, and MUSCLE. The package requires no additional software packages and runs on all major platforms. Moreover, the `msa` package provides an R interface to the powerful LaTeX package TeXshade [1] which allows for a highly customizable plots of multiple sequence alignments. Unless some very special features of TeXshade are required, users can pretty-print multiple sequence alignments without the need to know the details of LaTeX or TeXshade.

2 Installation

The `msa` `R` package (current version: 1.0.2) is available via Bioconductor. The simplest way to install the package is the following:

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("msa")
```

To test the installation of the `msa` package, enter

```
library(msa)
```

in your `R` session. If this command terminates without any error message or warning, you can be sure that the `msa` package has been installed successfully. If so, the `msa` package is ready for use now and you can start performing multiple sequence alignments.

To make use of all functionalities of `msaPrettyPrint()`, a TeX/LaTeX system [8] must be installed. To make use of LaTeX code created by `msaPrettyPrint()` or to use the output of `msaPrettyPrint()` in Sweave [10] or knitr [21] documents, the LaTeX package TeXshade (file `texshade.sty`) [1] must be accessible to the LaTeX system too. The file `texshade.sty` is shipped with the `msa` package. To determine where the file is located, enter the following command in your `R` session:

```
system.file("tex", "texshade.sty", package="msa")
```

Alternatively, TeXshade can be installed directly from the Comprehensive TeX Archive Network (CTAN).[1]

[1] `https://www.ctan.org/pkg/texshade`

3 msa for the Impatient

In order to illustrate the basic workflow, this section presents a simple example with default settings and without going into the details of each step. Let us first load amino acid sequences from one of the example files that are supplied with the msa package:

```
mySequenceFile <- system.file("examples", "exampleAA.fasta", package="msa")
mySequences <- readAAStringSet(mySequenceFile)
mySequences

##   A AAStringSet instance of length 9
##     width seq                                    names
## [1]   452 MSTAVLENPGLGRKLS...NSEIGILCSALQKIK PH4H_Homo_sapiens
## [2]   453 MAAVVLENGVLSRKLS...SEVGILCNALQKIKS PH4H_Rattus_norve...
## [3]   453 MAAVVLENGVLSRKLS...SEVGILCHALQKIKS PH4H_Mus_musculus
## [4]   297 MNDRADFVVPDITTRK...LNAGDRQGWADTEDV PH4H_Chromobacter...
## [5]   262 MKTTQYVARQPDDNGF...RLGLHAPLFPPKQAA PH4H_Pseudomonas_...
## [6]   451 MSALVLESRALGRKLS...SSEVEILCSALQKLK PH4H_Bos_taurus
## [7]   313 MAIATPTSAAPTPAPA...LNAGTREGWADTADI PH4H_Ralstonia_so...
## [8]   294 MSGDGLSNGPPPGARP...AYATAGGRLAGAAAG PH4H_Caulobacter_...
## [9]   275 MSVAEYARDCAAQGLR...VARRKDQKALDPATV PH4H_Rhizobium_loti
```

Now that we have loaded the sequences, we can run the msa() function which, by default, runs ClustalW with default parameters:

```
myFirstAlignment <- msa(mySequences)

## use default substitution matrix

myFirstAlignment

## CLUSTAL 2.1
##
## Call:
##    msa(mySequences)
##
## AAMultipleAlignment with 9 rows and 456 columns
##      aln                                    names
## [1] MAAVVLENGVLSRKLSDF...SINSEVGILCNALQKIKS PH4H_Rattus_norve...
## [2] MAAVVLENGVLSRKLSDF...SINSEVGILCHALQKIKS PH4H_Mus_musculus
## [3] MSTAVLENPGLGRKLSDF...SINSEIGILCSALQKIK- PH4H_Homo_sapiens
## [4] MSALVLESRALGRKLSDF...SISSEVEILCSALQKLK- PH4H_Bos_taurus
## [5] ------------------...GWADTEDV---------- PH4H_Chromobacter...
## [6] ------------------...GWADTADI---------- PH4H_Ralstonia_so...
## [7] ------------------...AYATAGGRLAGAAAG--- PH4H_Caulobacter_...
## [8] ------------------...------------------ PH4H_Pseudomonas_...
## [9] ------------------...------------------ PH4H_Rhizobium_loti
```

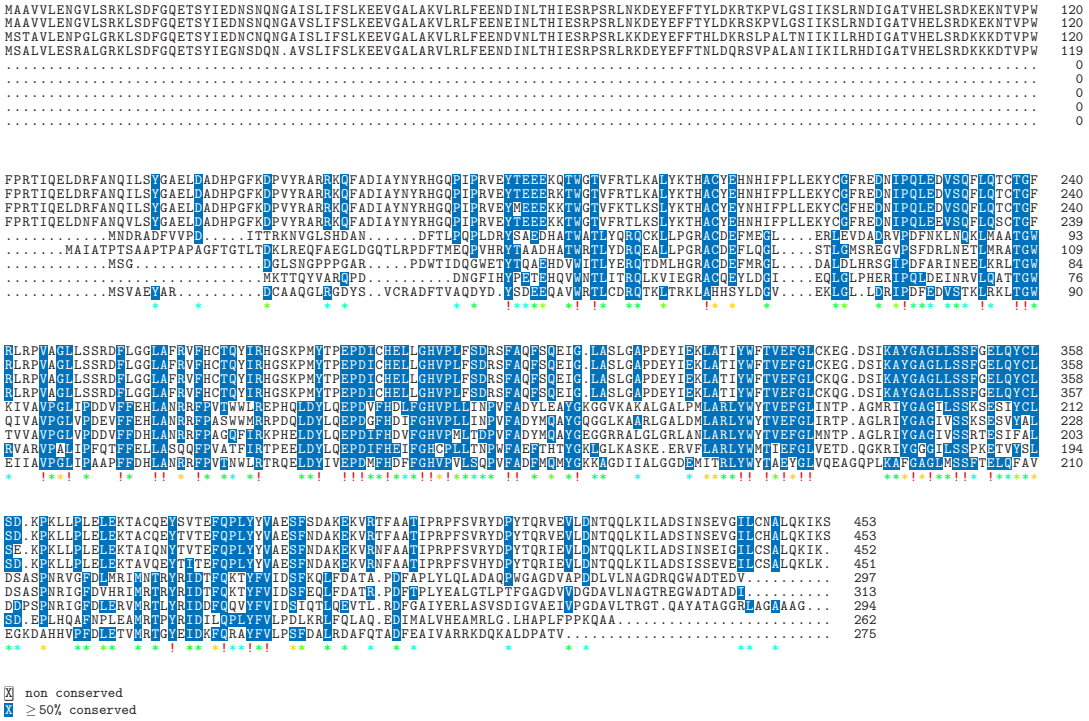


Figure 1: The PDF file `myfirstAlignment.pdf` created with `msaPrettyPrint()`.

The `msa` package offers the function `msaPrettyPrint()` which allows for pretty-printing multiple alignments using the LaTeX package TeXshade. As an example, the following `R` code creates a PDF file `myfirstAlignment.pdf` which is shown in Figure 1:

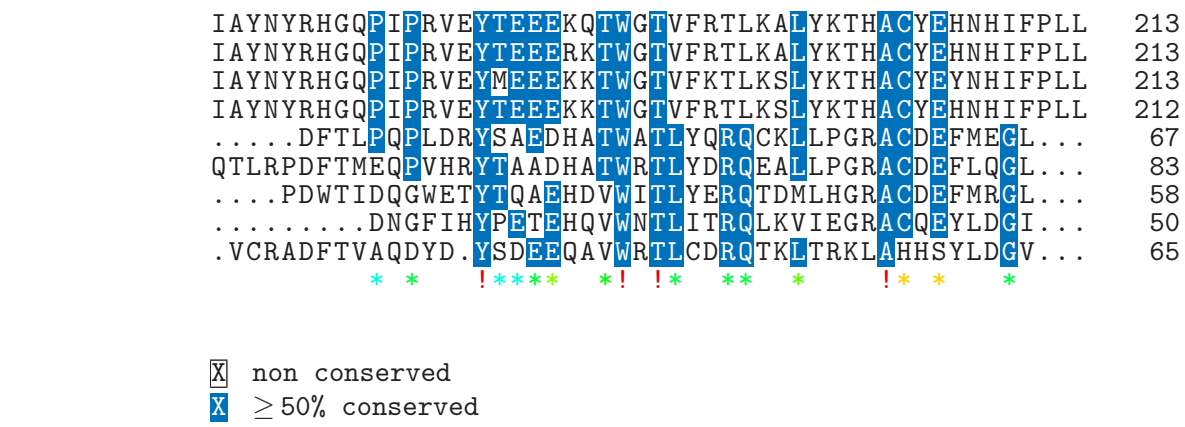
```
msaPrettyPrint(myFirstAlignment, output="pdf", showNames="none",
               showLogo="none", askForOverwrite=FALSE, verbose=FALSE)
```

In the above call to `msaPrettyPrint()`, the printing of sequence names has been suppressed by `showNames="none"`. The settings `askForOverwrite=FALSE` and `verbose=FALSE` are necessary for building this vignette, but, in an interactive `R` session, they are not necessary.

Almost needless to say, the file names created by `msaPrettyPrint()` are customizable. By default, the name of the argument is taken as file name. More importantly, the actual output of `msaPrettyPrint()` is highly customizable, too. For more details, see the Section 5 and the help page of the function (`?msaPrettyPrint`).

The `msaPrettyPrint()` function is particularly useful for pretty-printing multiple sequence alignments in Sweave [10] or knitr [21] documents. More details are provided in Section 5. Here, we restrict to a teasing example:

```
msaPrettyPrint(myFirstAlignment, y=c(164, 213), output="asis",
               showNames="none", showLogo="none", askForOverwrite=FALSE)
```



4 Functions for Multiple Sequence Alignment in More Detail

The example in Section 3 above simply called the function `msa()` without any additional arguments. We mentioned already that, in this case, ClustalW is called with default parameters. We can also explicitly request ClustalW or one of the two other algorithms ClustalOmega or Muscle:

```
myClustalWAlignment <- msa(mySequences, "ClustalW")

## use default substitution matrix

myClustalWAlignment

## CLUSTAL 2.1
##
## Call:
##    msa(mySequences, "ClustalW")
##
## AAMultipleAlignment with 9 rows and 456 columns
##      aln                                          names
## [1] MAAVVLENGVLSRKLSDF...SINSEVGILCNALQKIKS PH4H_Rattus_norve...
## [2] MAAVVLENGVLSRKLSDF...SINSEVGILCHALQKIKS PH4H_Mus_musculus
## [3] MSTAVLENPGLGRKLSDF...SINSEIGILCSALQKIK- PH4H_Homo_sapiens
## [4] MSALVLESRALGRKLSDF...SISSEVEILCSALQKLK- PH4H_Bos_taurus
## [5] ------------------...GWADTEDV---------- PH4H_Chromobacter...
## [6] ------------------...GWADTADI---------- PH4H_Ralstonia_so...
## [7] ------------------...AYATAGGRLAGAAAG--- PH4H_Caulobacter_...
## [8] ------------------...------------------ PH4H_Pseudomonas_...
## [9] ------------------...------------------ PH4H_Rhizobium_loti
```

```
myClustalOmegaAlignment <- msa(mySequences, "ClustalOmega")

## using Gonnet

myClustalOmegaAlignment

## ClustalOmega 1.2.0
##
## Call:
##    msa(mySequences, "ClustalOmega")
##
## AAMultipleAlignment with 9 rows and 467 columns
##      aln                                        names
## [1] MSALVLESRALGRKLSDF...SISSEVEILCSALQKLK- PH4H_Bos_taurus
## [2] MSTAVLENPGLGRKLSDF...SINSEIGILCSALQKIK- PH4H_Homo_sapiens
## [3] MAAVVLENGVLSRKLSDF...SINSEVGILCNALQKIKS PH4H_Rattus_norve...
## [4] MAAVVLENGVLSRKLSDF...SINSEVGILCHALQKIKS PH4H_Mus_musculus
## [5] ------------------...------------------ PH4H_Pseudomonas_...
## [6] ------------------...------------------ PH4H_Rhizobium_loti
## [7] ------------------...LAGAAAG----------- PH4H_Caulobacter_...
## [8] ------------------...V----------------- PH4H_Chromobacter...
## [9] ------------------...I----------------- PH4H_Ralstonia_so...

myMuscleAlignment <- msa(mySequences, "Muscle")
myMuscleAlignment

## MUSCLE 3.8.31
##
## Call:
##    msa(mySequences, "Muscle")
##
## AAMultipleAlignment with 9 rows and 460 columns
##      aln                                        names
## [1] MAAVVLENGVLSRKLSDF...SINSEVGILCNALQKIKS PH4H_Rattus_norve...
## [2] MAAVVLENGVLSRKLSDF...SINSEVGILCHALQKIKS PH4H_Mus_musculus
## [3] MSTAVLENPGLGRKLSDF...SINSEIGILCSALQKIK- PH4H_Homo_sapiens
## [4] MSALVLESRALGRKLSDF...SISSEVEILCSALQKLK- PH4H_Bos_taurus
## [5] ------------------...------------------ PH4H_Pseudomonas_...
## [6] ------------------...------------------ PH4H_Rhizobium_loti
## [7] ------------------...AYATAGGRLAGAAAG--- PH4H_Caulobacter_...
## [8] -----------MNDRADF...QGWADTEDV--------- PH4H_Chromobacter...
## [9] MAIATPTSAAPTPAPAGF...EGWADTADI--------- PH4H_Ralstonia_so...
```

Please note that the call `msa(mySequences, "ClustalW", ...)` is just a shortcut for the call `msaClustalW(mySequences, ...)`, analogously for `msaClustalOmega()` and `msaMuscle()`.

In other words, `msa()` is nothing else but a wrapper function that provides a unified interface to the three functions `msaClustalW()`, `msaClustalOmega()`, and `msaMuscle()`.

All three functions `msaClustalW()`, `msaClustalOmega()`, and `msaMuscle()` have the same parameters: The input sequences are passed as argument `inputSeqs`, and all functions have the following arguments: `cluster`, `gapOpening`, `gapExtension`, `maxiters`, `substitutionMatrix`, `order`, `type`, and `verbose`. The ways these parameters are interpreted, are largely analogous, although there are some differences, also in terms of default values. See the subsections below and the man page of the three functions for more details. All of the three functions `msaClustalW()`, `msaClustalOmega()`, and `msaMuscle()`, however, are not restricted to the parameters mentioned above. All three have a '`...`' argument through which several other algorithm-specific parameters can be passed on to the underlying library. The following subsections provide an overview of which parameters are supported by each of the three algorithms.

4.1 ClustalW-Specific Parameters

The original implementation of ClustalW offers a lot of parameters for customizing the way a multiple sequence alignment is computed. Through the '`...`' argument, `msaClustalW()` provides an interface to make use of most these parameters (see the documentation of ClustalW[2] for a comprehensive overview). Currently, the following restrictions and caveats apply:

- The parameters `infile`, `clustering`, `gapOpen`, `gapExt`, `numiters`, `matrix`, and `outorder` have been renamed to the standardized argument names `inputSeqs`, `cluster`, `gapOpening`, `gapExtension`, `maxiters`, `substitutionMatrix`, and `order` in order to provide a consistent interface for all three multiple sequence alignment algorithms.
- Boolean flags must be passed as logical values, e.g. `verbose=TRUE`.
- The parameter `quiet` has been replaced by `verbose` (with the exact opposite meaning).
- The following parameters are (currently) not supported: `bootstrap`, `check`, `fullhelp`, `interactive`, `maxseqlen`, `options`, and `tree`.
- For the parameter `output`, only the choice `"clustal"` is available.

4.2 ClustalOmega-Specific Parameters

In the same way as ClustalW, the original implementation of ClustalOmega also offers a lot of parameters for customizing the way a multiple sequence alignment is computed. Through the '`...`' argument, `msaClustalOmega()` provides an interface to make use of most these parameters (see the documentation of ClustalOmega[3] for a comprehensive overview). Currently, the following restrictions and caveats apply:

- The parameters `infile`, `clustersize`, `gapOpen`, `gapExt`, `iterations`, and `out-order` have been renamed to the argument names `inputSeqs`, `cluster`, `gapOpening`, `gapExtension`,

[2] `http://www.clustal.org/download/clustalw_help.txt`
[3] `http://www.clustal.org/omega/README`

`maxiters`, and `order` in order to provide a consistent interface for all three multiple sequence alignment algorithms.

- Boolean flags must be passed as logical values, e.g. `verbose=TRUE`.
- The following parameters are (currently) not supported: `maxSeqLength` and `help`.
- For the parameter `outFmt`, only the choice `"clustal"` is available.

4.3 MUSCLE-Specific Parameters

Finally, also MUSCLE offers a lot of parameters for customizing the way a multiple sequence alignment is computed. Through the '`...`' argument, `msaMuscle()` provides an interface to make use of most these parameters (see the documentation of MUSCLE[4] for a comprehensive overview). Currently, the following restrictions and caveats apply:

- The parameters `in`, `gapOpen`, `gapExtend`, `matrix`, and `seqtype` have been renamed to `inputSeqs`, `gapOpening`, `gapExtension`, `substitutionMatrix` and `type` in order to provide a consistent interface for all three multiple sequence alignment algorithms.
- Boolean flags must be passed as logical values, e.g. `verbose=TRUE`.
- The parameter `quiet` has been replaced by `verbose` (with the exact opposite meaning).
- The following parameters are currently not supported: `clw`, `clwstrict`, `fastaout`, `group`, `html`, `in1`, `in2`, `log`, `loga`, `msaout`, `msf`, `out`, `phyi`, `phyiout`, `phys`, `physout`, `refine`, `refinew`, `scorefile`, `spscore`, `stable`, `termgaps4`, `termgapsfull`, `termgapshalf`, `termgapshalflonger`, `tree1`, `tree2`, `usetree`, `weight1`, and `weight2`.

5 Pretty-Printing Multiple Sequence Alignments

As already mentioned above, the `msa` package offers the function `msaPrettyPrint()` which allows for pretty-printing multiple sequence alignments using the LaTeX package TeXshade [1]. Which prerequisites are necessary to take full advantage of the `msaPrettyPrint()` function is described in Section 2.

The `msaPrettyPrint()` function writes a multiple sequence alignment to an alignment (`.aln`) file and then creates LaTeX code for pretty-printing the multiple sequence alignment on the basis of the LaTeX package TeXshade. Depending on the choice of the `output` argument, the function `msaPrettyPrint()` either prints a LaTeX fragment to the `R` session (choice `output="asis"`) or writes a LaTeX source file (choice `output="tex"`) that it processes to a DVI file (choice `output="dvi"`) or PDF file (choice `output="pdf"`). Note that no extra software is needed for choices `output="asis"` and `output="tex"`. For `output="dvi"` and `output="pdf"`, however, a TeX/LaTeX distribution must be installed in order to translate the LaTeX source file into the desired target format (DVI or PDF).

[4] `http://www.drive5.com/muscle/muscle.html`

The function `msaPrettyPrint()` allows for making the most common settings directly and conveniently via an R interface without the need to know the details of LaTeX or TeXshade. In the following, we will describe some of these customizations. For all possibilities, the user is referred to the documentation of TeXshade.[5]

5.1 Consensus Sequence and Sequence Logo

The consensus sequence of the alignment is one of the most important results of a multiple sequence alignment. `msaPrettyPrint()` has a standard possibility to show this consensus sequence with the parameter `showConsensus`. The default value is `"bottom"`, which results in the following:

```
msaPrettyPrint(myFirstAlignment, output="asis", y=c(164, 213),
               subset=c(1:6), showNames="none", showLogo="none",
               consensusColor="ColdHot", showLegend=FALSE,
               askForOverwrite=FALSE)
```

Consensus sequences can also be displayed on top of a multiple sequence alignment or omitted completely.

In the above example, an exclamation mark '`!`' in the consensus sequence stands for a conserved letter, i.e. a sequence positions in which all sequences agree, whereas an asterisk '`*`' stands for positions in which there is a majority of sequences agreeing. Positions in which the sequences disagree are left blank in the consensus sequence. For a more advanced example how to customize the consensus sequence, see the example in Subsection 5.4 below.

The color scheme of the consensus sequence can be configured with the `consensusColors` parameter. Possible values are `"ColdHot"`, `"HotCold"`, `"BlueRed"`, `"RedBlue"`, `"GreenRed"`, `"RedGreen"`, or `"Gray"`. The above example uses the color scheme `"RedGreen"`.

Additionally, `msaPrettyPrint()` also offers a more sophisticated visual representation of the consensus sequence — sequence logos. Sequence logos can be displayed either on top of the multiple sequence alignment (`showLogo="top"`), below the multiple sequence alignment (`showLogo="bottom"`), or omitted at all (`showLogo="none"`):

```
msaPrettyPrint(myFirstAlignment, output="asis", y=c(164, 213),
               subset=c(1:6), showNames="none", showLogo="top",
               logoColors="rasmol", shadingMode="similar",
               showLegend=FALSE, askForOverwrite=FALSE)
```

[5] `https://www.ctan.org/pkg/texshade`

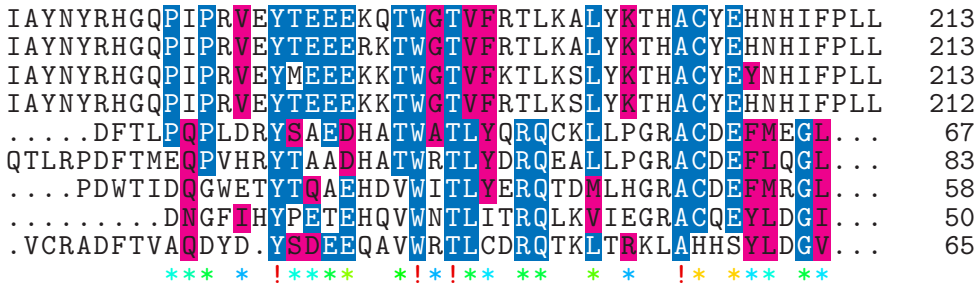
The color scheme of the sequence logo can be configured with the `logoColors` parameter. Possible values are `"chemical"`, `"rasmol"`, `"hydropathy"`, `"structure"`, `"standard area"`, and `"accessible area"`. The above example uses the color scheme `"rasmol"`.

Finally note that a consensus sequence and a sequence logo can be displayed together, but only on opposite sides.

5.2 Color Shading Modes

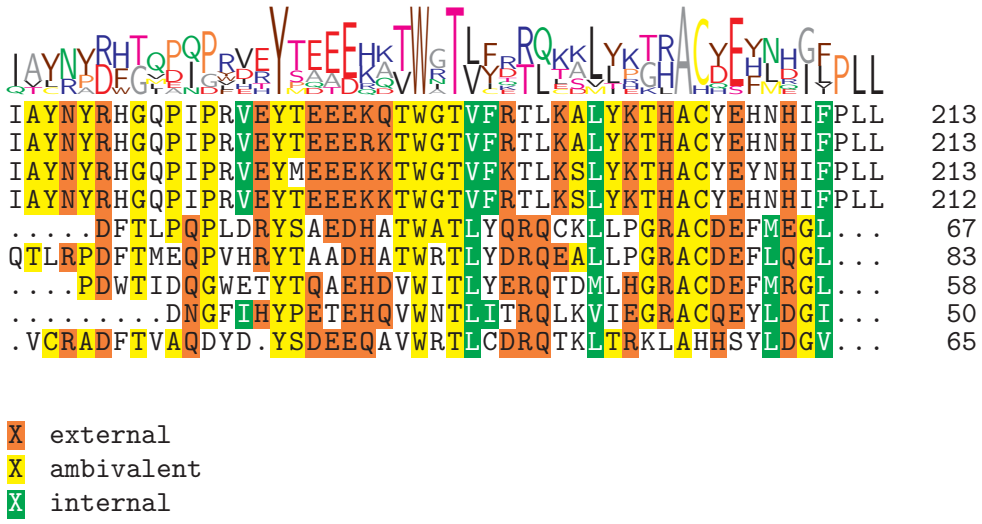
TEXshade offers different shading schemes for displaying the multiple sequence alignment itself. The following schemes are available: `"similar"`, `"identical"`, and `"functional"`. Moreover, there are five different color schemes available for shading: `"blues"`, `"reds"`, `"greens"`, `"grays"`, or `"black"`. The following example uses the shading mode `"similar"` along with the color scheme `"blues"`:

```
msaPrettyPrint(myFirstAlignment, output="asis", y=c(164, 213),
               showNames="none", shadingMode="similar",
               shadingColors="blues", showLogo="none",
               showLegend=FALSE, askForOverwrite=FALSE)
```



If the shading modes `"similar"` or `"identical"` are used, the `shadingModeArg` argument allows for setting a similarity threshold (a numerical value between 0 and 100). For shading mode `"functional"`, the following settings of the `shadingModeArg` argument are possible: `"charge"`, `"hydropathy"`, `"structure"`, `"hemical"`, `"rasmol"`, `"standard area"`, and `"accessible area"`. The following example uses shading mode `"functional"` along with `shadingModeArg` set to `"structure"`:

```
msaPrettyPrint(myFirstAlignment, output="asis", y=c(164, 213),
               showNames="none", shadingMode="functional",
               shadingModeArg="structure",
               askForOverwrite=FALSE)
```



In the above example, a legend is shown that specifies the meaning of the color codes with which the letters are shaded. In some of the other examples above, we have suppressed this legend with the option `showLegend=FALSE`. The default, however, is that a legend is printed underneath the multiple sequence alignment like in the previous example.

5.3 Subsetting

In case that not the complete multiple sequence alignment should be printed, `msaPrettyPrint()` offers two ways of sub-setting. On the one hand, the `subset` argument allows for selecting only a subset of sequences. Not surprisingly, `subset` must be a numeric vector with indices of sequences to be selected. On the other hand, it is also possible to slice out certain positions of the multiple sequence alignment using the `y` argument. In the simplest case, `y` can be a numeric vector with two elements in ascending order which correspond to the left and right bounds between which the multiple sequence alignment should be displayed. However, it is also possible to slice out multiple windows. For this purpose, the argument `y` must be an `IRanges` object containing the starts and ends of the windows to be selected.

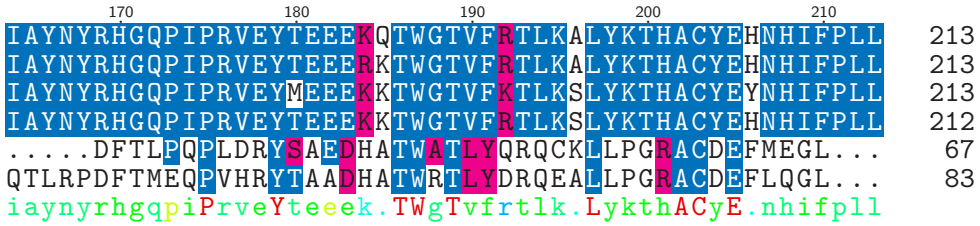
5.4 Additional Customizations

The `msaPrettyPrint()` function provides an interface to the most common functionality of TEXshade in a way that the user does not need to know the specific commands of TEXshade. TEXshade, however, provides a host of additional customizations many of which are not covered by the interface of the `msaPrettyPrint()` function. In order to allow users to make use of all functionality of TEXshade, `msaPrettyPrint()` offers the `furtherCode` argument through which users can add LATEX code to the `texshade` environment that is created by `msaPrettyPrint()`.

Moreover, the `code` argument can be used to bypass all of `msaPrettyPrint()`'s generation of TEXshade code.

Here is an example how to use the `furtherCode` argument in order to customize the consensus sequence and to show a ruler on top:

```
msaPrettyPrint(myFirstAlignment, output="asis", y=c(164, 213),
               subset=c(1:6), showNames="none", showLogo="none",
               consensusColor="ColdHot", showLegend=FALSE,
               shadingMode="similar", askForOverwrite=FALSE,
               furtherCode=c("\\defconsensus{.}{lower}{upper}",
                             "\\showruler{1}{top}"))
```



5.5 Sweave or knitr Integration

The function `msaPrettyPrint()` is particularly well-suited for pretty-printing multiple alignments in Sweave [10] or knitr [21] documents. The key is to set `output` to `"asis"` when calling `msaPrettyPrint()` and, at the same time, to let the `R` code chunk produce output that is directly included in the resulting LaTeX document as it is. This can be accomplished with the code chunk option `results="tex"` in Sweave and with the code chunk option `results="asis"` in knitr. Here is an example of a Sweave code chunk that displays a pretty-printed multiple sequence alignment inline:

```
<<AnyChunkName,results="tex">>=
msaPrettyPrint(myFirstAlignment, output="asis")
@
```

The same example in knitr:

```
<<AnyChunkName,results="asis">>=
msaPrettyPrint(myFirstAlignment, output="asis")
@
```

Note that, for processing the resulting LaTeX source document, the TEXshade package must be installed (see Section 2) and the TEXshade package must be loaded in the preamble:

```
\usepackage{texshade}
```

5.6 Further Caveats

- Note that `texi2dvi()` and `ttexi2pdf()` 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.
- TeXshade has a wide array of functionalities. Only the most common ones have been tested for interoperability with R. So the use of the arguments `furtherCode` and `code` is the user's own risk!

6 Known Issues

Memory Leaks

The original implementations of ClustalW, ClustalOmega, and MUSCLE are stand-alone command line programs which are only run once each time a multiple sequence alignment is performed. During the development of the `msa` package, we performed memory management checks using Valgrind [13] and discovered multiple memory leaks in ClustalW and MUSCLE. These memory leaks have no effect for the command line tools, since the program is closed each time the alignment is finished. In the implementation of the `msa` package, however, these memory leaks may have an effect if the same algorithm is run multiple times.

For MUSCLE, we managed to eliminate all memory leaks by deactivating the two parameters `weight1` and `weight2`. ClustalOmega did not show any memory leaks. ClustalW indeed has several memory leaks which are benign if the algorithm is run only a few times, but which may have more severe effects if the algorithm is run many times. ClustalOmega also has a minor memory leak, but the loss of data is so small that no major problems are to be expected except for thousands of executions of ClustalOmega.

ClustalOmega vs. Older GCC Versions on Linux/Unix

We have encountered peculiar behavior of ClustalOmega if the package was built using an older GCC version: if we built the package on an `x86_64` Linux system with GCC 4.4.7, ClustalOmega built smoothly and could be executed without any errors. However, the resulting multiple sequence alignment was more than sub-optimal. We could neither determine the source of this problem nor which GCC versions show this behavior. We therefore recommend Linux/Unix users to use an up-to-date GCC version (we used 4.8.2 during package development, which worked nicely) or, in case they encounter dubious results, to update to a newer GCC version and re-install the package.

ClustalOmega: OpenMP Support on Mac OS

ClustalOmega is implemented to make use of OpenMP (if available on the target platform) [4]. Due to issues on one of the Bioconductor build servers running Mac OS, we had to deactivate OpenMP generally for Mac OS platforms. If a Mac OS user wants to re-activate OpenMP, he/she should download the source package tarball, untar it, comment/uncomment the corresponding

line in `msa/src/ClustalOmega/msaMakefile` (see first six lines), and build/install the package from source.

MUSCLE with Custom Substitution Matrices

We are aware the that our MUSCLE interface is rather picky in terms of the format in which substitution matrices are passed to the `msaMuscle()` function. This interface will be improved in future versions.

7 Future Extensions

We envision the following changes/extensions in future versions of the package:

- Integration of more multiple sequence alignment algorithms, such as, T-Coffee [15] or others [3, 11, 12, 18]
- Support for retrieving guide trees from the multiple sequence alignment algorithms
- Interface to methods computing phylogenetic trees (e.g. as contained in the original implementation of ClustalW)
- Elimination of memory leaks described in Section 6 and re-activation of parameters that have been deactivated in order to avoid memory leaks
- More tolerant handling of custom substitution matrices (MUSCLE interface)

8 How to Cite This Package

If you use this package for research that is published later, you are kindly asked to cite it as follows [2]:

> U. Bodenhofer, E. Bonatesta, C. Horejš-Kainrath, and Sepp Hochreiter(2015). msa: an R Package for multiple sequence alignment. *Bioinformatics* (accepted). DOI: 10.1093/bioinformatics/btv494.

To obtain a BibTEX entries of the reference, enter the following into your R session:

```
toBibtex(citation("msa"))
```

Moreover, we insist that, any time you cite the package, you also cite the original paper in which the original algorithm has been introduced (see bibliography below).

9 Change Log

Version 1.0.2:

- fix of improperly aligned sequence logos produced by `msaPrettyPrint()`
- updated citation information

Version 1.0.1: fix of `msa()` function

Version 1.0.0: first official release as part of Bioconductor 3.1

References

[1] E. Beitz. TEXshade: shading and labeling of multiple sequence alignments using LATEX2e. *Bioinformatics*, 16(2):135–139, 2000.

[2] U. Bodenhofer, E. Bonatesta, C. Horejš-Kainrath, and S. Hochreiter. msa: an R package for multiple sequence alignment. *Bioinformatics*, 2015. (accepted).

[3] M. Brudno, M. Chapman, B. Göttgens, S. Batzoglou, and B. Morgenstern. Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics*, 4:66, 2003.

[4] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

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

[6] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32(5):1792–1797, 2004.

[7] R. C. Edgar and S. Batzoglou. Multiple sequence alignment. *Curr. Opin. Struct. Biol.*, 16(3):368–373, 2006.

[8] L. Lamport. *LATEX — A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley Longman, Amsterdam, 1999.

[9] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, and D. G. Higgins. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[10] F. Leisch. Sweave: dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580, Heidelberg, 2002. Physica-Verlag.

[11] A. Löytynoja, A. J. Vilella, and N. Goldman. Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. *Bioinformatics*, 28(13):1684–1691, 2012.

[12] B. Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3):211–218, 1999.

[13] N. Nethercote and J. Seward. Valgrind: A framework for heavyweight dynamic binary instrumentation. In *Proc. of the ACM SIGPLAN 2007 Conf. on Programming Language Design and Implementation*, San Diego, CA, 2007.

[14] C. Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput. Biol.*, 3(8):e123, 2007.

[15] C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, 302(1):205–217, 2000.

[16] H. Pagès, P. Aboyoun, R. Gentleman, and S. DebRoy. *Biostrings: String objects representing biological sequences, and matching algorithms*, 2015. R package version 2.36.1.

[17] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.*, 7:539, 2011.

[18] A. M. Szalkowski. Fast and robust multiple sequence alignment with phylogeny-aware gap placement. *BMC Bioinformatics*, 13:129, 2012.

[19] J. D. Thompson, D. G. Higgins, and T. J. Gibson. 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, 2004.

[20] I. M. Wallace, G. Blackshields, and D. G. Higgins. Multiple sequence alignments. *Curr. Opin. Struct. Biol.*, 15(3):261–266, 2005.

[21] Y. Xie. *Dynamic Documents with R and knitr*. Chapman & Hall/CRC, 2014.