Using `muscle` to produce multiple sequence alignments in `Bioconductor`

Alex T. Kalinka

October 17, 2016
alex.t.kalinka@gmail.com
Institute for Population Genetics, Vetmeduni Vienna, Veterinärplatz 1, 1210 Vienna, Austria.

Abstract

Producing high-quality multiple sequence alignments of DNA, RNA, or amino acid sequences is often an essential component of any comparative sequence-based study. The MUSCLE algorithm employs a progressive alignment approach to optimise pairwise alignment scores, and achieves both high accuracy and reduced computational time even when handling thousands of sequences (Edgar, 2004,a). The `R` package `muscle` integrates the MUSCLE algorithm into the `Bioconductor` project by utilizing existing `Biostrings` classes for representing sequence objects and multiple alignments.

Contents

1 Introduction

Performing multiple sequence alignments of biological sequences is often an essential aspect of studies that utilize sequence data. For example, multiple sequence alignments are at the core of several studies, such as phylogenetic tree estimation based on sequence data, testing for signatures of selection in coding or non-coding sequences, comparative genomics, secondary structure prediction, or critical residue identification. Hence, the multiple sequences may be homologous sequences belonging to several different species, paralogous sequences belonging to a single species, orthologous sequences belonging to multiple individuals of a single species, or any other variant thereof.

The MUSCLE algorithm is a progressive alignment method that works with DNA, RNA, and amino acid sequences producing high-accuracy alignments with very fast computational times (Edgar, 2004,a). The algorithm is iterative, with later iterations refining the earlier alignments. In each iteration, pairwise alignment scores are computed for all sequence pairs (based on k-mer counting or global pairwise alignments) and the values are entered into a triangular distance matrix. This matrix is then used to build a binary tree of all the sequences (using one of various different hierarchical clustering algorithms, such as UPGMA or neighbour-joining). A progressive alignment is then built from this matrix by following the tree from the tips (individual sequences) to the root (all sequences aligned) adding in gaps as appropriate.

2 Example session

First, we must load the `muscle` package into our current `R` session:

```
> library(muscle)
```

To illustrate the package, we will perform a multiple sequence alignment of the MAX gene (Wagner et al., 1992) across 31 mammalian species. These sequences are available in the `umax` object that is part of the `muscle` package, and is an object of class `DNAStringSet`:

```
> umax
```

```
  A DNAStringSet instance of length 31
     width seq                                                names
 [1]   483 ATGAGCGATAACGATGACATCGA...GCTCCGGATGGAGGCCAGCTAA Ailuropoda_melano...
 [2]   489 ATGAGCGATAACGATGACATCGA...GCTCCGGATGGAGGCCAGCTAA Bos_taurus
 [3]   483 ATGAGCGATAACGATGACATCGA...GCTCCGGATGGAGGCCAGCTAA Callithrix_jacchus
 [4]   483 ATGAGCGATAACGATGACATCGA...GCTCCGGATGGAGGCCAGCTAA Canis_familiaris
 [5]   483 ATGAGCGATAACGATGACATCGA...ACTCCGGATGGAGGCCAGCTAA Cavia_porcellus
 ...   ... ...
[27]   483 ATGAGCGATAACGATGACATCGA...ACTGCGGATGGAGGCCAGCTAA Rattus_norvegicus
[28]   483 ATGAGCGATAACGATGACATCGA...GCTCCGGATGGAGGCCAGCTAA Sorex_araneus
[29]   447 GAAGAGCATCCGAGGTTTCAATC...GCTTCGGATGGAGACCAGCTAA Tarsius_syrichta
[30]   444 GAAGAGCAACCGAGGTTTCAATC...ACTCCGCATGGAGGCCAGCTAA Tupaia_belangeri
[31]   447 GAAGAGCAACCGAGGTTTCAATC...GCTCCGGATGGAGGCCAGCTAA Tursiops_truncatus
```

All input to the `muscle` function should be objects of class `XStringSet`, which can be one of `DNAStringSet`, `RNAStringSet`, or `AAStringSet` (see package Biostrings (Pages et al., 2015)). An alignment is generated as follows (`muscle` automatically detects whether the input is DNA, RNA, or amino acid):

```
> aln <- muscle(umax)
```

The output is an object of class `MultipleAlignment` (see package Biostrings):

```
> aln
```

```
DNAMultipleAlignment with 31 rows and 492 columns
      aln                                                names
 [1] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Ailuropoda_melano...
 [2] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Bos_taurus
 [3] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Callithrix_jacchus
 [4] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Canis_familiaris
 [5] ATGAGCGATAACGATGACATCGAGG...GAAACTCCGGATGGAGGCCAGCTAA Cavia_porcellus
 [6] ATGAGCGATAACGATGACATCGAGG...GAAACTCCGGATGGAGGCCAGCTAA Choloepus_hoffmanni
 [7] ATGAGCGATAACGATGACATCGAGG...GAAACTCCGGATGGAGGCCAGCTAA Dipodomys_ordii
 [8] ATGAGCGATAACGATGACATCGAGG...GAAACTCCGCATGGAGGCCAGCTAA Echinops_telfairi
 [9] -------------------------...GAAGCTCCTGATGGAGGCCAGCTAA Erinaceus_europaeus
 ... ...
[23] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Sus_scrofa
[24] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Pongo_abelii
[25] -------------------------...GAAACTCCGGATGGAGGCCAGCTAA Procavia_capensis
[26] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGAGTGGAGGCCAGCTAA Oryctolagus_cunic...
[27] ATGAGCGATAACGATGACATCGAGG...GAAACTGCGGATGGAGGCCAGCTAA Rattus_norvegicus
[28] ATGAGCGATAACGATGACATCGAGG...GAAGCTCCGGATGGAGGCCAGCTAA Sorex_araneus
[29] -------------------------...GAAGCTTCGGATGGAGACCAGCTAA Tarsius_syrichta
[30] -------------------------...GAAACTCCGCATGGAGGCCAGCTAA Tupaia_belangeri
[31] -------------------------...GAAGCTCCGGATGGAGGCCAGCTAA Tursiops_truncatus
```

If the desired input is initially present in an external file, such as a `fasta` file, then these sequences can be read into an `XStringSet` object using one of the `XstringSet` input-output functions (`readDNAStringSet`, `readRNAStringSet`, or `readAAStringSet`). For example, to read in one of the example `fasta` files in the external data contained in the `Biostrings` package:

```
> file.path <- system.file("extdata", "someORF.fa", package = "Biostrings")
> orf <- readDNAStringSet(file.path, format = "fasta")
```

This will read in a `DNAStringSet` object containing 7 unaligned sequences:

```
> orf
```

```
  A DNAStringSet instance of length 7
    width seq                                               names
[1]  5573 ACTTGTAAATATATCTTTTATTT...CTTATCGACCTTATTGTTGATAT YAL001C TFC3 SGDI...
[2]  5825 TTCCAAGGCCGATGAATTCGACT...AGTAAATTTTTTTCTATTCTCTT YAL002W VPS8 SGDI...
[3]  2987 CTTCATGTCAGCCTGCACTTCTG...TGGTACTCATGTAGCTGCCTCAT YAL003W EFB1 SGDI...
[4]  3929 CACTCATATCGGGGGTCTTACTT...TGTCCCGAAACACGAAAAAGTAC YAL005C SSA1 SGDI...
[5]  2648 AGAGAAAGAGTTTCACTTCTTGA...ATATAATTTATGTGTGAACATAG YAL007C ERP2 SGDI...
[6]  2597 GTGTCCGGGCCTCGCAGGCGTTC...AAGTTTTGGCAGAATGTACTTTT YAL008W FUN14 SGD...
[7]  2780 CAAGATAATGTCAAAGTTAGTGG...GCTAAGGAAGAAAAAAAAATCAC YAL009W SPO7 SGDI...
```

3 Arguments for the `muscle` function

Many different arguments can be passed to the `muscle` function, and these are described in detail in the online documentation. These arguments are either options (taking various values) or flags (either `TRUE` or `FALSE`). Here, I describe some of the more commonly-used arguments.

Enhanced speed. To enhance the speed of the algorithm, the `diags = TRUE` flag will optimize the speed with a potential loss of accuracy:

```
> aln <- muscle(umax, diags = TRUE)
```

Gap penalties. Default gap penalties can be modified to produce altered alignments. The gap penalty must be negative, with larger negative values indicating more stringent penalties:

```
> aln <- muscle(umax, gapopen = -30)
```

Remove progress indicators. When running the algorithm repeatedly (for a batch of sequences, for example), it may be preferred to stop output of the algorithm's progress to the screen (e.g. if there is a global progress indicator running):

```
> aln <- muscle(umax, quiet = TRUE)
```

Maximum number of hours. If an alignment is expected to take a long time, a maximum total number of hours can be specified, which, if reached, will lead to the algorithm stopping at this point and returning the current alignment:

```
> aln <- muscle(umax, maxhours = 24.0)
```

Log file. To find out what default settings are being used for all the arguments, a log file can be written to disk using the `log` argument in conjunction with the verbose argument, e.g. `log = "log.txt", verbose = TRUE`. This will write out the default values to the file `log.txt` in the current working directory of `R`.

4 R Session Information

The examples in this vignette were run under the following conditions:

```
> sessionInfo()

R version 3.3.1 (2016-06-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.1 LTS

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] muscle_3.16.0       Biostrings_2.42.0   XVector_0.14.0
[4] IRanges_2.8.0       S4Vectors_0.12.0    BiocGenerics_0.20.0

loaded via a namespace (and not attached):
[1] zlibbioc_1.20.0 tools_3.3.1
```

References

Edgar, R. C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, **32**, 1792-1797.

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

Pages, H., Aboyoun, P., Gentleman, R. and DebRoy, S. (2015) Biostrings: String objects representing biological sequences, and matching algorithms. R package version 2.34.1

Wagner, A. J., et al. (1992) Expression, regulation, and chromosomal localization of the Max gene. *Proc Natl Acad Sci USA*, **89**, 3111-3115.