Improving trimAl ability to cope with heterogeneous multiple sequence alignments.

Víctor Fernández-Rodríguez[#1,], Toni Gabaldón[#3,4,5], Salvador Capella-Gutierrez[#2]

[#1,2]*Life Sciences Department, Barcelona Supercomputing Center, Barcelona, Spain.*
[1]victor.fernandez@bsc.es, [2]salvador.capella@bsc.es
[#3]*Bioinformatics and Genomics Programme, Centre for Genomic Regulation (CRG), Barcelona, Spain*
[#4]*Universitat Pompeu Fabra (UPF), Barcelona, Spain*
[#5]*Institucio Catalana de Recerca I Estudis Avançats (ICREA), Barcelona, Spain*
[3]toni.gabaldon@crg.eu

Keywords - Multiple Sequence Alignment, Alignment filtering, Phylogenetic analysis

Extended Abstract

Alignments of biological sequences, called Multiple Sequence Alignments (MSA), are the entrypoint for many biological applications including evolutionary studies. However, the current algorithms used to reconstruct them tend to minimize (or maximize) mathematical functions rather than truly representing biological events. This is especially relevant for highly variable sequences regions where the positional homology is difficult to infer. This often tends to produce MSAs with a high noise-to-signal ratio, which will be eventually amplificated on downstream analyses that rely on them.

Thus, MSAs refinement has become a common practice in many biological domains. However, MSAs refinement algorithms are not except of errors so further investigation is needed making this area a very active research field.

Here we present a revisited version of trimAl, a popular resource aiming to improve MSAs using manual and/or automated methods. We will explain why is important to refactor trimAl's source code including issues found and solutions applied. Finally we will introduce a set of new functionality only achieved after improving the existing source code.

A. *Introduction*

trimAl was born as a internal laboratory script, that grew fast in functionality and code length. The original code was written in C, and later moved to C++ to exploit the Object Oriented Programming Paradigm (OOP).

The fast growth of the code, due to addition of new functionality and lack of a project pre-production phase led to a fully functional and almost bug-free but coupled code with some evident issues.

For this reason, in the present document we explain some relevant aspects of the refactoring step performed and the results obtained through the process.

B. *Format Machine State*

We have implemented a machine state to load and save MSAs in different formats, which allows to isolate the format handling code from the rest of the program.

This new paradigm allows to remove and/or implement new format handlers with ease, and also, allows the community to provide their own format handlers.

C. *Memory Improvement*

The original implementation loaded into memory a copy of the complete MSA each time any operation was applied. This leads to have a high degree of redundancies among loaded copies e.g. sequences names, metadata, etc. Indeed, up to three MSAs containing subsets of the same information are allocated at the same time in memory: original alignment, so we can compare the result obtained with the original; current alignment, the one that is being processed at the current step, and the resulting alignment.

In the new implementation, we followed a different memory management strategy, at a potential cost of performance.

We have the data on memory once, and all copies would point to that information and contain a pair of vectors indicating if we would reject or keep specific columns and/or sequences

D. *Speed Improvement*

One of the effects that highly coupled code had on the original implementation was that some statistics were computed more than once, increasing the time needed to perform an analysis.

The new approach allowed us to detect and avoid repeating calculations, and thus, reduce the time needed to perform the same analysis, with the same results.

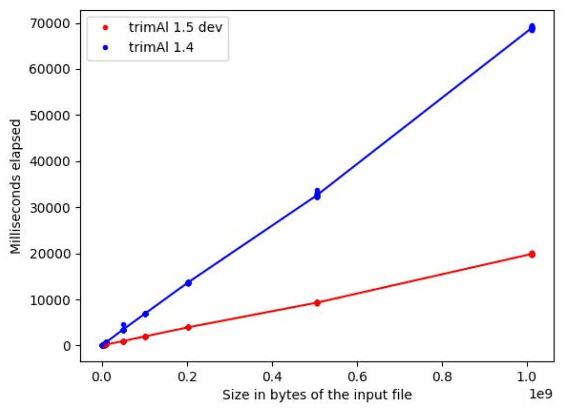


Fig.1. Time needed by the original and new implementations of trimAl using the strict algorithm (the most consuming of the program).

E. *Reporting Improvement*

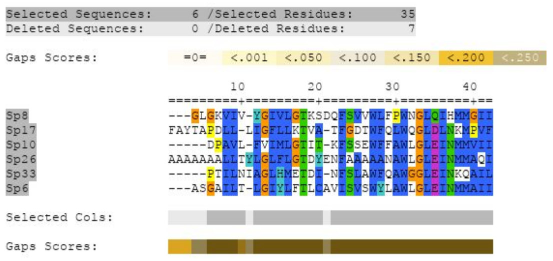
Reporting has been improved in several ways: Statistics report has been eased visually, using the correct tabulation and adding a header specifying the statistic being reported and the original filename of the alignment which it is extracted from.

More relevant is the new format for trimming reporting: The original implementation outputted an HTML file with a graphic visualization of the results of the trimming steps and the statistics used to perform these steps.

This allows the user to have an insight of what was removed and why.

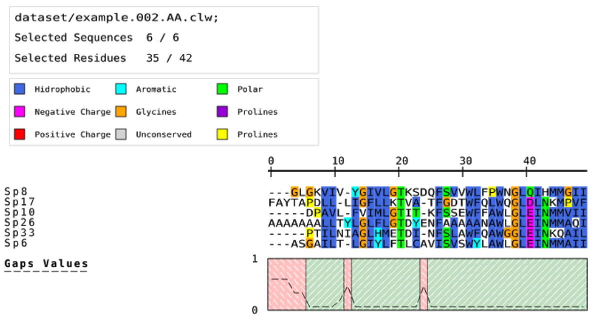
The new format, SVG, allows faster load, and better representation of the statistics, using a graph approximation, where the original used categories.

This lead to a more informative reporting, and also, more useful, as the report can be treated as a vectorial image, allowing to cut and scale it as much as needed.



```
Selected Sequences:      6 /Selected Residues:      35
Deleted Sequences:       0 /Deleted Residues:        7

Gaps Scores:          =0=  <.001 <.050 <.100 <.150 <.200 <.250

                               10        20        30        40
                      =========+=========+=========+=========+==
Sp8                   ---GLGKVIV-YGIVLGTKSDQFSVVWLFPWNGLQIHMMGII
Sp17                  FAYTAPDLL-LIGFLLKTVA-TFGDTWFQLWQGLDLNKMPVF
Sp10                  -----DPAVL-FVIMLGTIT-KFSSEWFFAWLGLEINMMVII
Sp26                  AAAAAAALLTYLGLFLGTDYENFAAAAANAWLGLEINMMAQI
Sp33                  -----PTILNIAGLHMETDI-NFSLAWFQAWGGLEINKQAIL
Sp6                   ---ASGAILT-LGIYLFTLCAVISVSWYLAWLGLEINMMAII

Selected Cols:

Gaps Scores:
```

Fig.2. HTML version of the trimming report.



```
dataset/example.002.AA.clw;
Selected Sequences   6 / 6
Selected Residues   35 / 42
```

🟦	Hidrophobic	🟦	Aromatic	🟩	Polar
🟪	Negative Charge	🟧	Glycines	🟪	Prolines
🟥	Positive Charge	⬜	Unconserved	🟨	Prolines

```
                       0         10        20        30        40
Sp8                    ---GLGKVIV-YGIVLGTKSDQFSVVWLFPWNGLQIHMMGII
Sp17                   FAYTAPDLL-LIGFLLKTVA-TFGDTWFQLWQGLDLNKMPVF
Sp10                   -----DPAVL-FVIMLGTIT-KFSSEWFFAWLGLEINMMVII
Sp26                   AAAAAAALLTYLGLFLGTDYENFAAAAANAWLGLEINMMAQI
Sp33                   -----PTILNIAGLHMETDI-NFSLAWFQAWGGLEINKQAIL
Sp6                    ---ASGAILT-LGIYLFTLCAVISVSWYLAWLGLEINMMAII
Gaps Values          1

                     0
```

Fig.3. SVG version of the trimming report.

F. *Error Reporter*

Centralization and standardization of the errors and warnings that we provide to the users was required, as some problems had arisen from the lack of them.

These problems include reporting the same error with different messages, which would lead to confusion to the final user or having to do code scraping to find all the calls to an error message that we would like to change.

An Error Reporter has been created, isolating the code of error reporting from the rest of the code.

This allowed us to create a numbered list of errors, that allows to a better understanding of the situations that may arise from the use of the program.

It also allowed us to add a verbose option, allowing for better reporting control to the end user.

G. *Time Tracker*

To have a better understanding of the flow performed by the program, an auxiliary class has been implemented: the Time Tracker.

This class tracks the calls to most of the methods in the program, and outputs a tree where we can easily understand which methods calls to others in a specific execution, and calculate the time each method lasts, including and excluding calls to other tracked methods.

This allows us to see if the program behaves exactly as expected, and to pinpoint which methods are candidates to optimization.

This functionality has been enhanced by adding the ability to track memory before and after each method call. This allows to have a better understanding of the memory management on each method and globally.

H. *Conclusion and Future Enhancement*

Short-term future foresight includes containerization of the binaries, a complete revamp of the suite website and the extension to support Next Generation Sequencing (NGS) data. In the long run, we will deeply analyse the existing trimming algorithms to propose new ones, which can cope with MSAs made up to ten of thousands of sequences. Any newly developed method will be extensively benchmark to ensure their scientific and technical relevance for current and future end-users.

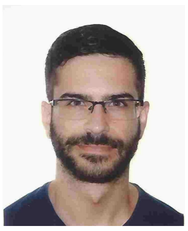
References

[1] Capella-Gutiérrez, S. & Gabaldón, T. Measuring guide-tree dependency of inferred gaps in progressive aligners. Bioinformatics 29, 1011–1017 (2013).

[2] Capella-Gutiérrez, S., Silla-Martínez, J. M. & Gabaldón, T. trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. Bioinforma. Appl. NOTE 25, 1972–197310 (2009).

[3] Dessimoz, C. & Gil, M. Phylogenetic assessment of alignments reveals neglected tree signal in gaps. Genome Biol. 11, R37 (2010).

Author biography



Víctor Fernández was born in Valencia, Spain, in 1992. He studied Biology in Universitat de València, Burjassot, where he developed an strong interest in genetics. After his degree, he worked on a startup developing video games, with the intention of improving his programming abilities to be able to obtain the maximum possible of his master's degree in bioinformatics.

Since November 2016 he has been enrolled in the Bioinformatics and Biological Computation Master degree at the Escuela Nacional de Sanidad (Madrid), and since July, he is part of the team developing the new version of trimAl at the BSC.