Reducing noise in protein multialignments
DD2404 Applied Bioinformatics

Maria Gerontini
Msc. Machine Learning
mger@kth.se

Moa Ristner
Msc. Machine Learning
moar@kth.se

KTH - December 15, 2012

Abstract

Multiple alignments consist a common way to detect evolutionary relationships between proteins and are a central subject on the bioinformatics field. It has been proved that a good quality of these alignments can improve the subsequent analyses for future experiments. In our approach we created a simple but effective algorithm for eliminating noisy columns and boost further analyses such as the phylogenetic reconstruction. Our goal was to remove regions that are not inherited and should therefore not be aligned and other regions that may have evolved fast and the current multialignment is impossible to infer. We compared the reduced trees and the original trees with the reference trees. From this we could conclude that it is worthwhile to reduce the noise.

1 Introduction

The alignment quality plays a significant role on the phylogenetic reconstruction [2] [1] and is high importance issue on Bioinformatics. Many researchers intend that by removing parts of the multialignments we remove useful information. Our goal was to provide an algorithm which can prove that by applying noise reduction algorithms on the alignments we can get the same results for the inferred trees and we can improve the performance of the phylogenetic methods.

2 Methodology

In order to solve the problem with the noisy protein multialignments we constructed an implementation using the python language. We considered the noisy columns based on three criteria.

- There are more than 50% indels.
- At least 50% of amino acids are unique.

- No amino acids appear more than two times.

The word indel is used to describe insertions or deletions on the protein sequence. These indels are represented with the symbol '-'. We used each column from the multialignments and we checked the occurrences for the specific characters that are represented on every column by using a dictionary provided by the module python 'collections'. If the occurrence of the symbol which represents the indels were more than 50% we consider this column noisy and we extract and we saved the number of this column in a specific list with the noisy column indices. The same applied for the next two criteria. From the same dictionary we get the occurrences for each amino acid if the columns contain at least 50% unique amino acids either the amino acids have occurrence between 1 and 2 then we classify that column as noisy.

Finally, in order to get the non noisy sequences through a loop we remove the noisy columns by using the list with the noisy column indices. The new data is saved to a new directory for further analysis.

In order to be able to use all the test data from the provided directories we used the python modules glob and os which allow to use pattern matching methods to go through all the directories from a given path.

To check for the correctness and the validity of our program we added some additional test data and more checking controls. Our program can treat effectively erroneous data files such us not fasta format,empty files, short sequences, only noisy columns etc. In addition to this, we populate our own testing file to be sure that our noise reduction algorithm works properly. The results are shown in Figure 1 where we have 4 different multialignments and the red are the noisy columns. Our program returned a list with all the noisy columns which is [5, 6, 7, 8, 9, 10, 11, 12, 13, 14] and it is the expected result.

A	A	A	A	A	B	B	B	B	B	C	C	C	C	C	K	K	K	K	K	L	L	L	L	L	J	J	J	J	J
A	A	A	A	A	C	C	C	C	C	-	-	-	-	-	K	K	K	K	K	L	L	L	L	L	J	J	J	J	J
A	A	A	A	A	D	D	D	D	D	-	-	-	-	-	K	K	K	K	K	P	P	P	P	P	J	J	J	J	J
A	A	A	A	A	I	I	I	I	I	-	-	-	-	-	K	K	K	K	K	P	P	P	P	P	I	I	I	I	I
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Figure 1: Noisy columns in a fasta file.

In order to check whether or not removing the noise has an impact on phylogeny interference we create a testing suite. Here we want to create an inferred tree from all of the alignments in the subdirectories. Both for the original ones and the new

ones which has had noise removed. This tree is then compared to the corresponding reference tree by calculating the symmetric distance.

The implementation is done using the python language. For each alignment the distance matrix is build by using the method fastprot. This distance matrix is then used as an input to fnj to build the corresponding phylogenetic tree.

To be able to run fastprot and fnj from within the python code we use the module subprocess. The resulting tree, in newick format, is then compared to the reference tree using both the method symmetric_difference and robinson_foulds_distance, both available within the DenroPy package.

The symmetric_difference method returns the number of splits that occur in one of the trees but not the other. In other words if this number is low, then the trees are more alike. The robinson-foulds method returns the sum of the square of differences in branch lengths for equivalent splits between the two trees ref

As we did in the noise reduction algorithm we use the glob and os modules to be able to read from all files in the sub-directories.

3 Results

The result from testing the reduction method can be seen in Figure 2, Figure 3, Figure 4 and Figure 5. Figure 2 shows the average symmetric distance for the different data sets. The orange bars are represent the average symmetric distance for the alignments which have had the noise removed. The blue bars are the original alignments. We can note that the orange bars are all lower than the blue bars.

In Figure 3 the percentage of the reduced symmetric distance is shown. Here we can more clearly see that there is indeed a shorter distance for the alignments with reduced noise. And we can also see a trend among the different alignments, both in regard of amount of mutations and depending on the alignment being symmetric or asymmetric.

As stated in the previous section we also tried to use the Robinson-Foulds metric available in the DendroPy package. The result can be seen in Figure 4.

We also looked at the distribution within each alignment, shown in Figure 5. This figure shows the distribution for the asymmetric_2.0 dataset for both the original alignment (blue color) and for the new alignment (red color). The distributions for the other sets can be found in Appendix A. The figure clearly shows that the distribution of the new alignments are shifted slightly to the left.

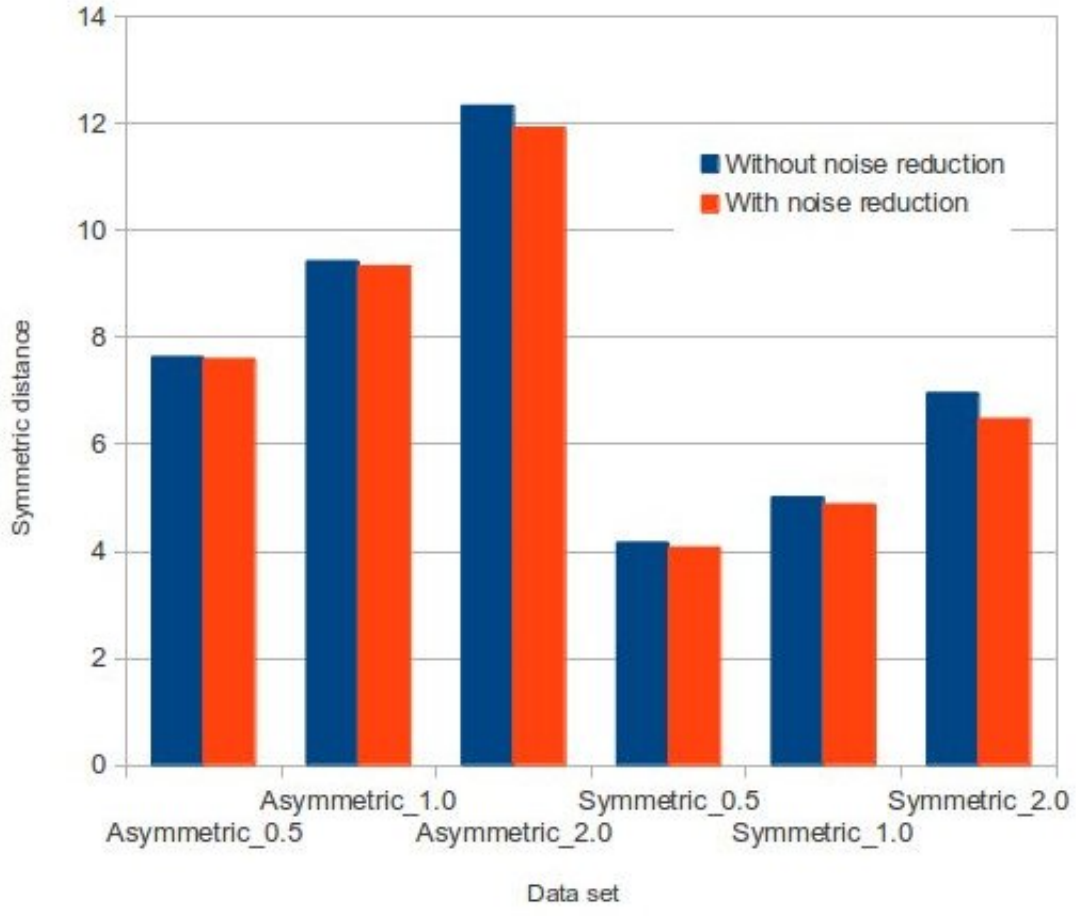

Figure 2: Average symmetric distance for the different alignments.

4 Conclusion and Discussion

One thing we can notice from Figure 2 is that, independent of the implementation of noise reduction, there are two clear trends. One of these is that the phylogenetic trees inferred from the symmetric alignments are more alike the reference trees than the asymmetric alignments. This is due to the asymmetric trees being more complex and therefore it is more likely that the inferred trees lies further away from the reference trees. The other trend shows that the alignments with more mutations (xx_0.5 has the lowest amount of mutations and xx_2.0 the highest) a

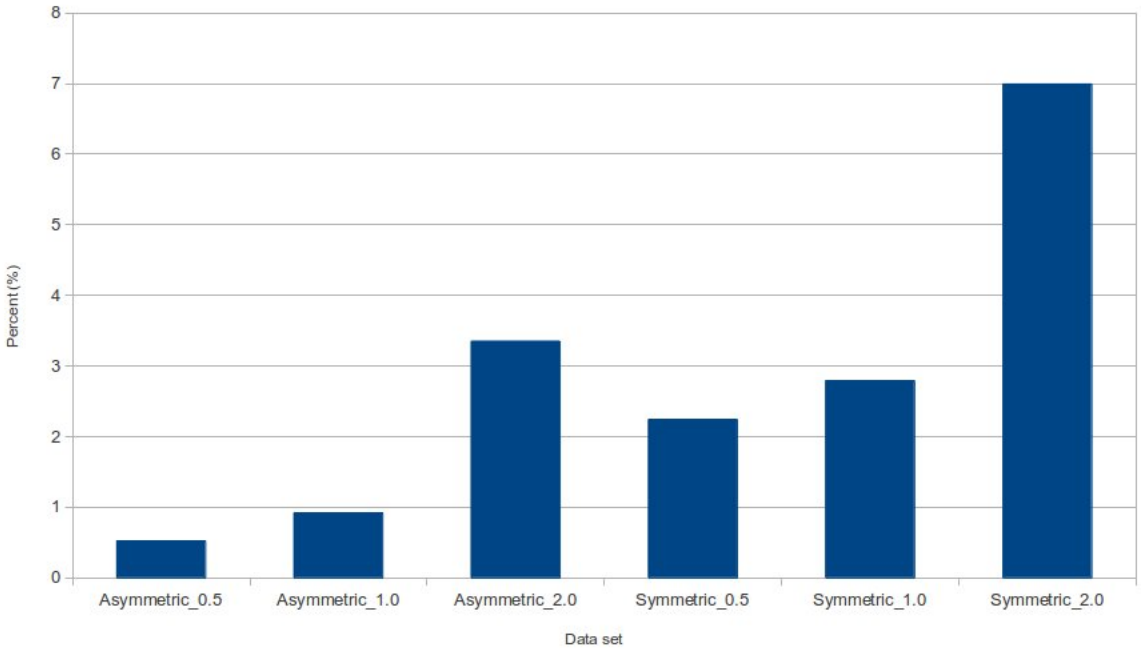

Figure 3: Percentage of reduced symmetric distance for the different alignments.

higher symmetric distance between the trees. This is what one would expect since the mutations causes a deviation from the reference tree.

The Robinson-Fould method, generating the plot in Figure 4 does not seem to result in any relevant information. It seems here that the distance is equal for almost all alignments and that the only thing that differs is whether or not the alignments are symmetric. But it is stated on the course web page that this method is claimed to be wrong, so we ignore this result.

From Figure 3 we can see that the noise reduction works best for the symmetric alignments, in particular for the symmetric_2.0 alignment. But we can see a positive effect of the noise reduction on all alignments, as we could see from Figure 2 as well. Therefore we can conclude that it is worthwhile to use multialignment noise reduction.

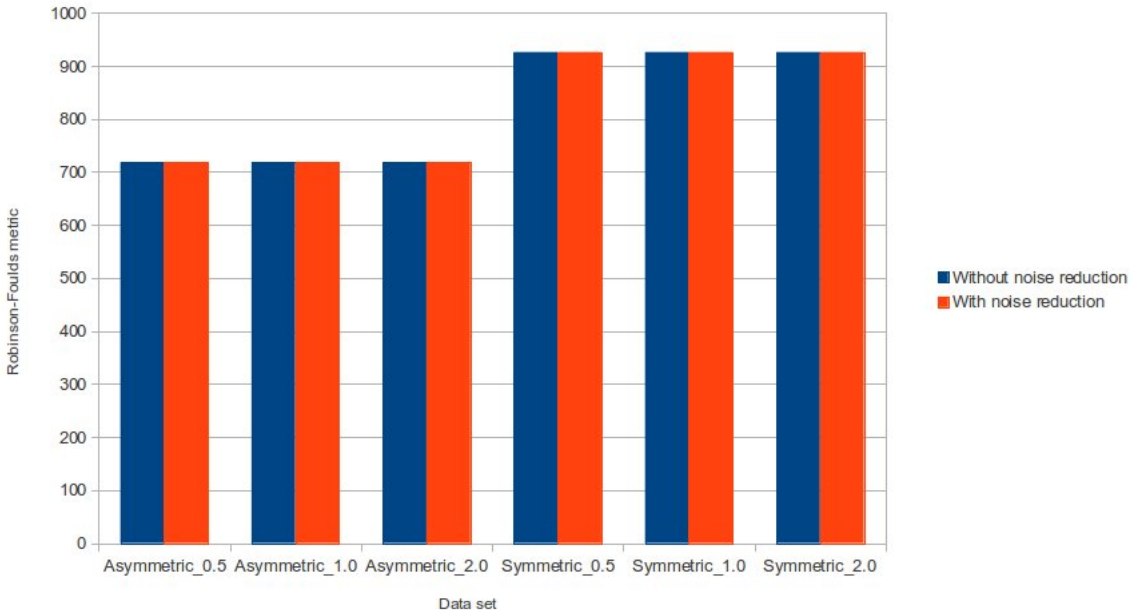

Figure 4: The average Robinson-Foulds distance for the different alignments.

A Appendix

References

[1] S. Capella-Gutiérrez, J.M. Silla-Martínez, and T. Gabaldón. trimal: a tool for automated alignment trimming in large-scale phylogenetic analyses. *Bioinformatics*, 25(15):1972–1973, 2009.

[2] G. Talavera and J. Castresana. Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments. *Systematic Biology*, 56(4):564–577, 2007.

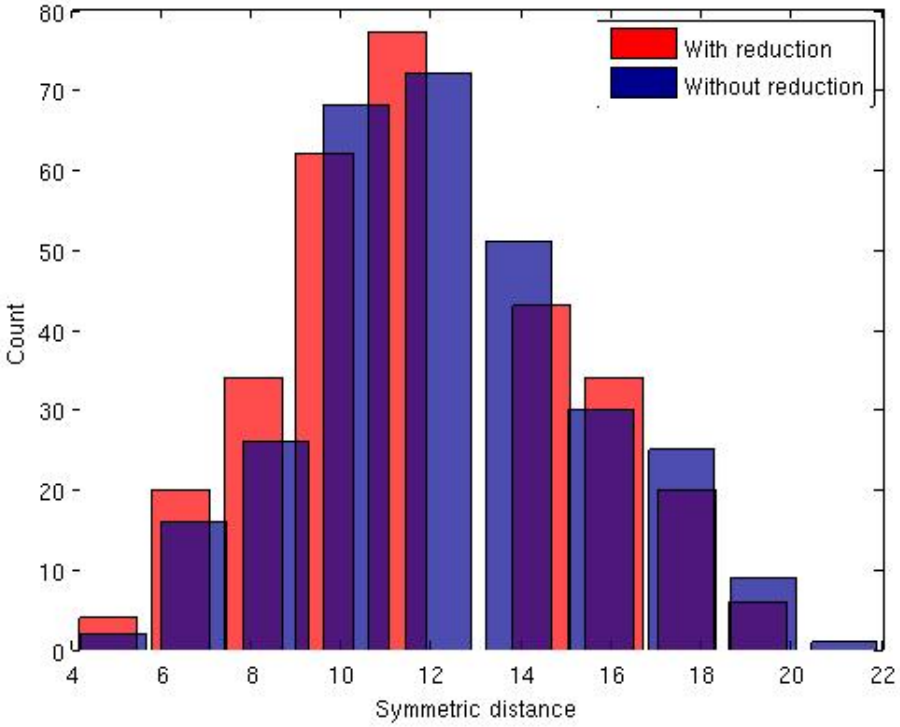

Figure 5: The distribution within the asymmetric_2.0 alignment for the original and the reduced alignment.

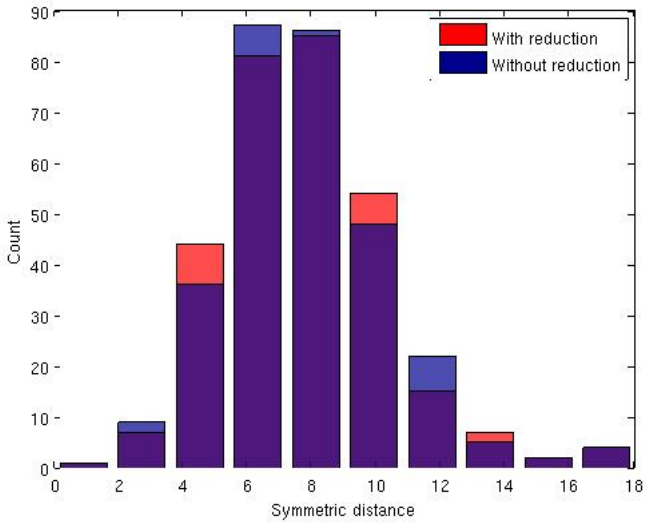

Figure 6: The distribution within the asymmetric_0.5 alignment for the original and the reduced alignment.

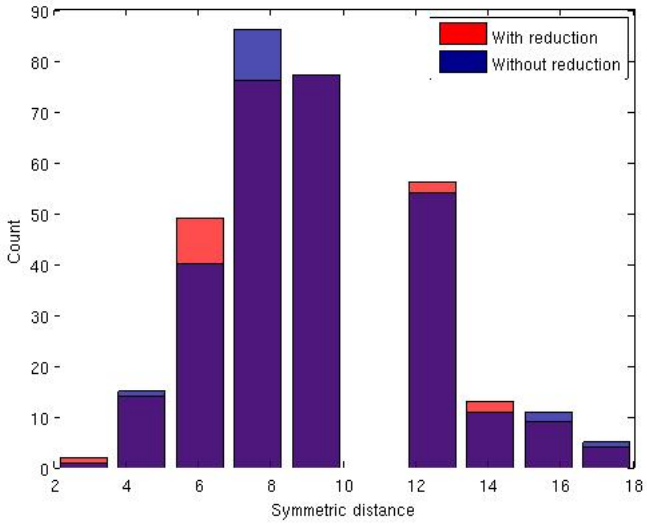

Figure 7: The distribution within the asymmetric_1.0 alignment for the original and the reduced alignment.

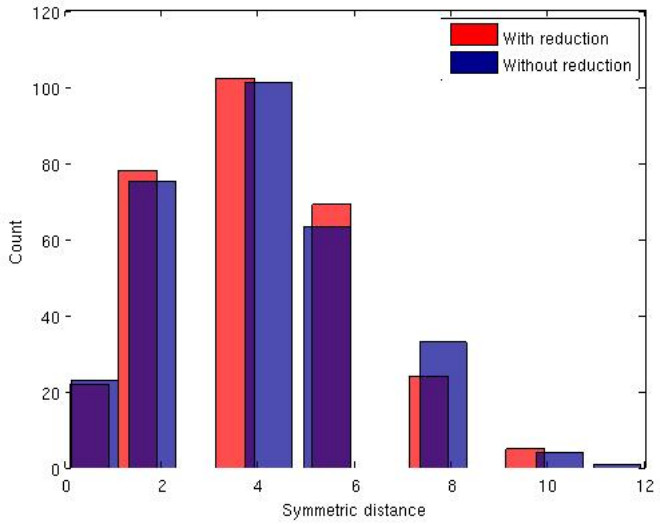

Figure 8: The distribution within the symmetric_0.5 alignment for the original and the reduced alignment.

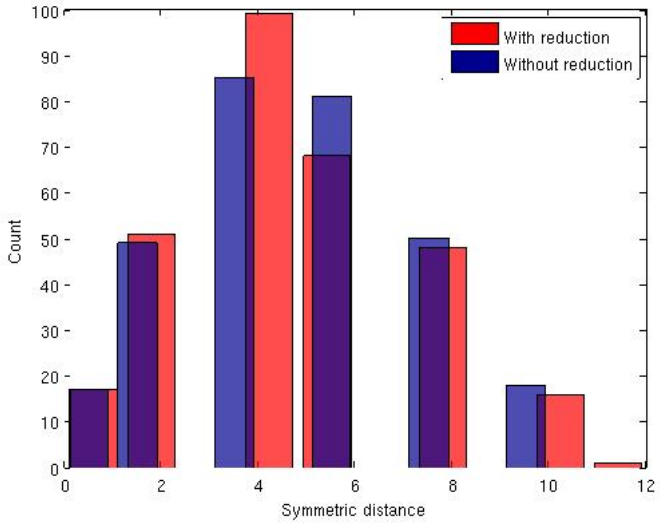

Figure 9: The distribution within the symmetric_1.0 alignment for the original and the reduced alignment.

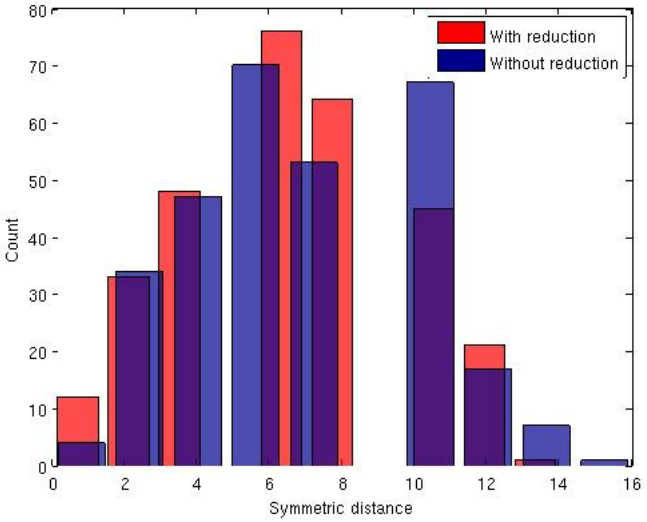

Figure 10: The distribution within the symmetric_2.0 alignment for the original and the reduced alignment.