Applied Bioinformatics DD2404 - Reducing noise in protein multialignments

Annika Bendes, abendes@kth.se

17/12 2017

We chose the project “Reducing noise in protein multialignments”, and tried to make a mind map of what should be included in the project. After that we sketched a flowchart where we tried to map out how the noise reducing function should be approached in code. We also listed other elements, such as an amino acid dictionary, that was necessary in order to be able to use the noise reduction function. Our plan is to make a function that takes a column from the sequences as a string, and calls for another function that checks if the column fulfils the “noisy criteria”. If the condition is met the second function will return True. If the second function returns false, the column stored, and later added to the output file.

In the following step, we started writing the code for the noise reduction part. In this stage, we decided to read the input files directly from the code and used a small test data that we had created ourselves in order to be able to fast and easy observe the output from the program.

We also made the dictionary containing all the amino acids. We think it will be necessary to have this dictionary to be able to apply the noise conditions.

We made a counting function (count_character) to count occurences of the different amino acids in the input file’s aligned columns.

The next steps in the noise-reduction part:

- Write code for the noise conditions in the same definition and make them True if the condition is met.

18/12 2014

We wrote the noise-reduction conditions in one function, and we tested the noise-reduction process on test files that we used yesterday, and the functions seems to be working. The columns in the multialignments are removed if the column fulfils either of the following criteria:

- there are more than 50% indels,
- at least 50% of amino acids are unique,
- no amino acid appears more than twice.

We then made control experiments where we tested and took care of the cases where 1) the file was not being able to open correctly, 2) if all columns are noisy (all of them have been removed), and 3) If the input file is empty. We fixed so that if any of these cases should appear, the program should exit with an error massage telling the problem.

The current program reads an infile, removes noisy columns and writes out the sequences without the noise columns. We still need to make some corrections (e.g. fix the fasta format of the output) before we can move to the next part of the project, inferring trees.

19/12 2014

Today, we made some corrections of the look of the output from the program. Names of the sequences were added and the output is now written in a FASTA-format. We also made a control for if the column contains a character which is not an amino acid. We decided to remove the column

containing characters that are not amino acids, and inform the user by printing a notification about it.

20/12 2014

We fixed a “head program” that takes the input files and calls for the previously made noise reduction functions. We also changed the structure so that the noise reduced files reside in a new folder.

18/10 2015:

We wanted to get a deeper understanding of how to implement tools from dendropy and the fastphylo tools fastprot and fnj, so we tried to call for dendropy in the terminal window. But we were unable to load dendropy and the KTH computers. We tried to install dendropy using

sudo pip install -U dendropy

But our user did not have permission to install dendropy.

Our supervisor gave us a solution to create a viritualenv (https://virtualenv.readthedocs.io/en/latest/userguide/#usage), using the following code:

```
mkdir lib
virtualenv ~/lib/virtual_python_envs
source ~/lib/virtual_python_envs/bin/activate
 pip install dendropy
```

18/9 2016:

Finally finished with my master thesis project, I decided to try to finish this project. We created a *viritalenv* using the code above, and we decided to now change the program so that we could run it from the terminal window using infiles = sys.argv[1] instead of specifying the files directly in the code. However, we realised that we had to add biopython to our viritual environment using

pip install biopython

in order for the code to work. First we tested the program with our small test file huppa.msl, and after that we used just the symmetric_0.5 file (containing 300 alignmnets) to only test a small portion of our data. The test resulted in a “results directory” containing the noise reduced files.

We tried to run fastprot and fnj on the computer, but they were not found on the KTH computers.

19/9 2016:

I tried to install fastphylo on my private windows compute, but fastprot did not come with the installation.

26/9 2016:

We started to write the introduction to the report.

28/9 2016

We got help from our professor, and fastprot and fnj could be called by writing

```
module use /misc/info/DD2404/appbio16/module
module add appbio16
```

instead of

module use /misc/info/DD2404/appbio15/module
module add appbio15

29/10 2016:

I first wanted to get an understanding about how the fastprot and fnj functions work, so we used the functions on a file in the terminal window. Here, we only got values when we used reconstruction method BIONJ for fnj, so BIONJ was later used in our script.

In accordance to Stafford Noble, we created functions in the script that called for fastprot and fnj so that we could avoid editing intermediate files.

At first when I tried to use fastprot on our data, we got an error text telling us that the format of the file was wrong. So I realised that our noise reduced files were not written out in the correct FASTA format that could be used in fastprot. So we had to rewrite the code and use AlignIO.read. for reading the input file that we later send to the noise reduction function. This solved our problems and we tested the program (with fastprot and fnj function) on a subset of the data files that we had.

However, I received the problem that we had too many temp files open at the same time when we tried to run the program for all of the data sets, so the process could not continue for all of them. To solve this problem, we had to add code for closing the temp trees after comparing them to the reference trees.

31/10 2016:

I started to analyse the data obtained from the previous run, when I realized that we had specified one of the noisy column criteria wrong in the code. Instead of removing columns if there are more than 50% indels, we removed the column if it has 50% or more indels. So instead of an if-statement where float(indels/len(column)) > 0.5, we had an if statement saying if float(indels/len(column)) >= 0.5. So I decided to correct this error to see how it would affect the results.

I also want to try to use the FNJ function with -m FNJ instead of BIONJ, since I wanted the distances to be unweighted.

4/11 2016:

We changed the FNJ function to use -m FNJ instead of BIONJ, and it appears that FNJ works now after we changes the FASTA format of our files.

I also started to analyse the results, and made barplots of the mean difference between the produced trees (aligned and noise reduced) and the reference tree.

16/11 2016

I wrote the Project setup part of the report and calculated the failure and success rate of the noise reduction (i.e. if the symmetric difference for the reference tree and the noise reduced tree were larger or less than for the unmodified tree and the reference tree).

I also calculated the frequency of reference tree being recovered for the unmodified files, and the noise reduced files.

17/11 2016

I wrote the results and conclusion part of the report.

28/11 2016

I wrote the Abstract to the report.

4/12 2016

We finished up the report and uploaded everything on Github.