Laboratory 8: multiple sequence alignment

This series of laboratory exercises attempts to align, using a variety of algorithms, (1) a divergent set of of nrDNA sequences (difficult) and (2) a set of relatively variable protein coding sequences (challenging).

In plants, nrDNA loci consist of many repeats. Each repeat is composed of ETS, 18S, ITS1, 5.8S, ITS2, and 26S. The sequences that you will work with today are 5.8S and ITS2 (with a very small portion of 26S). The 5.8S region is highly conserved and easily aligned whereas the ITS2 region is less conserved and therefore more difficult to align. The nrDNA sequences are from a study of cycad phenology (Griffith et al. 2012).

The protein coding sequences are Cupressaceae *ycf1*. Although *ycf1* serves an important function—part of the TIC complex that transports proteins synthesized in the cytosol through the innermost of plastid's two membranes—its sequence is highly variable and therefore challenging to align.

Tasks

(1) Install a few useful sequence analysis tools into your 'scripts' directory (created and added to your PATH as part of Laboratory 2).

   (a) Move to your 'scripts' directory by typing `cd scripts` in the terminal.

   (b) Download alan—a DNA and AA sequence wrapper for less—by typing `wget https://raw.githubusercontent.com/mpdunne/alan/master/alan` in the terminal.

   (c) Download a annotation filter for NCBI eTools by typing `wget https://raw.githubusercontent.com/dpl10/phytoinformatics2023/master/ftFilter.py` in the terminal.

   (d) Download a FASTA GenBank cleaning utility by typing `wget https://raw.githubusercontent.com/dpl10/phytoinformatics2023/master/GenBank2fasta.py` in the terminal.

   (e) Download an alignment scoring utility by typing `wget https://raw.githubusercontent.com/dpl10/phytoinformatics2023/master/pairs.py` in the terminal.

   (f) Download a FASTA filtering utility by typing `wget https://raw.githubusercontent.com/dpl10/phytoinformatics2023/master/selectFromFASTA.py` in the terminal.

   (g) Download a DNA/AA alignment tool (Abascal et al. 2010) by typing `wget https://raw.githubusercontent.com/dpl10/phytoinformatics2023/master/translatorX.pl` in the terminal.

   (h) Make the scripts executable by typing `chmod 0755 alan ftFilter.py GenBank2fasta.py pairs.py selectFromFASTA.py translatorX.pl` in the terminal.

   (i) Return to your home directory by typing `cd` in the terminal.

(2) Retrieve the nrDNA sequence data from GenBank:

   (a) Type `esearch -db nuccore -query 'internal AND transcribed AND Cycadidae AND Griffith NOT mitochondrial' | efetch -format fasta > nrITS-GenBank.fasta` in the terminal.

   (b) Confirm that 'nrITS-GenBank.fasta' has 38 sequences using grep.

(c) Clean up the GenBank FASTA format by typing `GenBank2fasta.py -f nrITS-GenBank.fasta > nrITS.fasta` in the terminal.

(d) Examine the files using alan by typing `alan nrITS-GenBank.fasta` and `alan nrITS.fasta` in the terminal (use the less command 'q' to exit alan, use arrows and/or page keys to scroll in less). Answer question (1).

(3) Install the alignment program MUSCLE (Edgar 2004a,b) by typing `sudo apt install muscle` in the terminal. Type your password when prompted. Make sure that your system is updated as well (use `sudo apt update && sudo apt upgrade && sudo apt autoremove` as needed).

(4) Align the sequences with MUSCLE by typing `muscle -in nrITS.fasta -out muscle-nrITS.fasta` in the terminal.

(5) View the alignment in alan by typing `alan muscle-nrITS.fasta` in the terminal. Answer question (2).

(6) Install the alignment program MAFFT (Katoh et al. 2002; Katoh & Toh 2008) from source.

(a) Download the compressed source code by typing `wget https://mafft.cbrc.jp/alignment/software/mafft-7.505-with-extensions-src.tgz` in the terminal.

(b) Expand the tar ball by typing `tar xvzf mafft-7.505-with-extensions-src.tgz` in the terminal.

(c) Change to the newly expanded directory by typing `cd mafft-7.505-with-extensions/` in the terminal.

(d) View the 'readme' file using the viewer of your choice (e.g. less). This file contains instructions for building and installing MAFFT.

(e) Change to the 'core' directory by typing `cd core/` in the terminal.

(f) Compile the program using the make script by typing `make -j$(nproc) CFLAGS="-march=native -O2"` in the terminal (there will be a fair number of warnings, but no errors).

(g) Install the newly compiled software by typing `sudo make install` in the terminal. Type your password if prompted.

(h) Change to the 'extensions' directory by typing `cd ../extensions/` in the terminal.

(i) Compile the extensions by typing `make -j$(nproc) CFLAGS="-march=native -O2"` in the terminal (again there will be warnings, but no errors).

(j) Install the extensions by typing `sudo make install` in the terminal. Type your password if prompted.

(k) Exit the MAFFT directory by typing `cd` in the terminal.

(l) Read the on–line MAFFT instructions (https://mafft.cbrc.jp/alignment/software/algorithms/algorithms.html) to learn how MAFFT works and the less informative man page to determine command–line options. Answer question (3).

(7) Align the sequences using the default settings in MAFFT by typing `mafft --auto nrITS.fasta > mafft-nrITS-auto.fasta` in the terminal. Open the alignment in alan and answer question (4).

(8) Compare the two alignments by typing `pairs.py -f muscle-nrITS.fasta` and `pairs.py -f mafft-nrITS-auto.fasta` in the terminal. Answer question (5).

(9) Align the sequences using an iteratively refined global alignment by typing `mafft --retree 100 --maxiterate 1000 --fmodel --globalpair nrITS.fasta > mafft-nrITS-it.fasta` in the terminal. Answer question (6).

(10) The optional parameters '–ep' and '–op' can be used to further control the alignment. Generally ep is set to a value lower than op and op is set somewhere between 1 and 3. Experiment with these parameters by trying at least ten different combinations (e.g. type `mafft --retree 100 --maxiterate 1000 --fmodel --globalpair --ep 1.00 --op 0.00 nrITS.fasta > mafft-nrITS-it10.fasta` in the terminal to set ep to one and op to zero). Use pairs.py to compare the alignments using the automatic alignment as the reference (e.g. type `pairs.py -f mafft-nrITS-it10.fasta` in the terminal). Fill in the table below and answer question (7).

–ep	–op	sum of pairs

(11) Align the sequences using an iteratively refined local alignment by typing `mafft --retree 100 --maxiterate 1000 --fmodel --localpair nrITS.fasta > mafft-nrITS-il.fasta` in the terminal. Answer question (8).

(12) Difficult to align sequences can often be better aligned if additional structural data are used. In the case of protein coding sequences, aligning nucleotides in amino acid space often produces a better alignment (i.e. translating the nucleotides to amino acids, aligning the amino acids, back translating the amino acids to nucleotides). In the case of rDNA there are no amino acids that can be used, but the secondary structure of the rDNA may be used. Secondary structure alignment can be very slow, so first select a representative subset of sequences to align (one of each genus).

   (a) Create a inclusion list by typing `grep -e GQ203984.1 -e GQ203988.1 -e GQ203989.1 -e GQ203992.1 -e GQ203994.1 -e GQ203996.1 -e GQ204004.1 -e GQ204009.1 -e GQ204010.1 -e GQ204017.1 nrITS.fasta > include` in the terminal.

   (b) Extract the exemplar sequences by typing `selectFromFASTA.py -f nrITS.fasta -l include > skeleton.fasta` in the terminal.

   (c) Create a file of the remaining sequences by typing `selectFromFASTA.py -i -f nrITS.fasta -l include > remainders.fasta` in the terminal.

   (d) Using grep, confirm that 'skeleton.fasta' and 'remainders.fasta' have exactly 38 sequences in total and none of them are repeated.

(13) To produce a 'fast' structural alignment, type `mafft-xinsi --retree 100 --maxiterate 1000 --fmodel skeleton.fasta > skeleton-structure.fasta` in the terminal. Answer question (9).

(14) Add the remaining sequences to the structural alignment by typing `mafft --maxiterate 1000 --seed skeleton-structure.fasta remainders.fasta > mafft-nrITS-structure.fasta` in the terminal. Answer question (10).

(15) Retrieve the Cupressaceae *ycf1* protein coding sequences from GenBank:

   (a) Download by typing `esearch -db nuccore -query 'refseq[filter] AND "complete genome" AND plastid[filter] AND "Cupressaceae"[Organism]' | efetch -format ft | ftFilter.py -g ycf1 | xargs -0 -I {} -P 1 bash -c 'ACCESSION=$(echo "{}" | awk "{print \$1}"); START=$(echo "{}" | awk "{print \$2}"); STOP=$(echo "{}" | awk "{print \$3}"); esearch -db nuccore -query "$ACCESSION" | efetch -format fasta -seq_start $START -seq_stop $STOP' > ycf1-GenBank.fasta` in the terminal. Answer question (11).

   (b) Confirm that 55 sequences were downloaded using grep.

   (c) Clean up the GenBank FASTA format using GenBank2fasta.py. Save the cleaned sequences in a file named 'ycf1.fasta'.

(16) Align the *ycf1* nucleotide sequences using MUSCLE. Answer question (12).

(17) Align the nucleotide sequences as amino acids using MUSCLE and translatorX by typing `translatorX.pl -i ycf1.fasta -o ycf1-muscleX.fasta -p M -c 11 -t T` in the terminal. Answer question (13).

(18) In order to use 'non–supported' alignment programs with translatorX, one must run translatorX to create the amino acid file (like in step (17)), run the alternative alignment program (or manually align), and then run translatorX again to back translate the amino acids to nucleotides. For example, to use KALIGN (Lassmann & Sonnhammer 2005; Lassmann 2020) and translatorX:

   (a) Install KALIGN by typing `sudo apt install kalign` in the terminal. Enter your password when prompted.

   (b) Align the amino acid sequences by typing `kalign -i ycf1-muscleX.fasta.aa_ali.fasta -o ycf1-kalign.fasta` in the terminal.

   (c) Create a nucleotide file for translatorX by typing `perl -pe 'if($_=~m/^[^>]/){$_=~tr/-//d}' ycf1-muscleX.fasta.nt_ali.fasta > ycf1-x.fasta` in the terminal. Answer question (14).

   (d) Convert the amino acid alignment to a nucleotide alignment by typing `translatorX.pl -i ycf1-x.fasta -a ycf1-kalign.fasta -o ycf1-kalignX.fasta -p M -c 11 -t T` in the terminal. Answer question (15).

Questions (https://forms.gle/8rP6zhhUQtVm9YRY9)

(1) For task (2)(d), what did 'GenBank2fasta.py' do to the downloaded file?

(2) For task (5):

   (a) Which sequences are poorly aligned?

   (b) What features indicate a poor alignment?

   (c) Which groups of sequences align well to one another, but not to other groups?

(d) How can you determine if these poorly aligned sequences should be alignable to the others?

(3) For task (6)(l), why did you install MAFFT from source code rather than using the Ubuntu 'mafft' package (hint: compare the files that you installed to the files listed in the Ubuntu package [https://packages.ubuntu.com/])?

(4) For task (7):

(a) How does the alignment subjectively compare to that of MUSCLE?

(b) Are the sequences that you identified as poorly aligned, better, worse, or the same?

(5) For task (8):

(a) Are the alignments different?

(b) For the calculation of sum of pairs, does it matter which values are used for the compatible cost and the incompatible cost?

(6) For task (9), what does each of the MAFFT parameters do?

(7) For task (10):

(a) Which setting worked the best? Why?

(b) Are the sequences that you identified as poorly aligned, better, worse, or the same?

(8) For task (11):

(a) Are the sequences that you identified as poorly aligned, better, worse, or the same?

(b) Are the sequences the same length before and after alignment?

(c) Indicate the commands that you used to determine this.

(9) For task (13):

(a) What does 'mafft-xinsi' do?

(b) Are the sequences that you identified as poorly aligned, better, worse, or the same?

(c) Are there other structural alignment options available?

(d) Are they better or worse in terms of alignment quality?

(10) For task (14):

(a) How does the alignment compare to a true multiple sequence alignment?

(b) What happens if the references ('skeleton-structure.fasta') are not completely representative in terms of length or sequence diversity?

(11) For task (15)(a), explain what each step of the command is doing.

(12) For task (16):

(a) Which sequences are poorly aligned?

(b) What features indicate alignment quality?

(13) For task (17):

(a) What does each of the translatorX options do?

(b) How does the translated alignment compare to nucleotide alignment?

(c) Are the sequences the same length before and after alignment?

(14) For task (18)(c):

(a) Explain what each step of the command is doing.

(b) Why is this step necessary?

(c) What happens if 'ycf1-muscleX.fasta.nt_ali.fasta' is used instead of 'ycf1-x.fasta'?

(15) For task (18)(d), what is the difference among the MUSCLE alignment, the MUSCLE/translatorX alignment, and the KALIGN/translatorX alignment? (Hint: compare the output of pairs.py [-a] for the MUSCLE and KALIGN nucleotide and amino acid alignments.)

Literature cited

Abascal, F., R. Zardoya & M. J. Telford. 2010. TranslatorX: multiple alignment of nucleotide sequences guided by amino acid translations. Nucleic Acids Research 38: W7–W13.

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

———. 2004b. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research 32: 1792–1797.

Griffith, M. P., M. A. Calonje, D. W. Stevenson, C. E. Husby & D. P. Little. 2012. Time, place, and relationship: cycad phenology in phylogenetic and biogeographic context. Memoirs of The New York Botanical Garden 106: 59–81.

Katoh, K., K. Misawa, K. Kuma & T. Miyata. 2002. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Research 30: 3059–3066.

Katoh, K. & H. Toh. 2008. Recent developments in the MAFFT multiple sequence alignment program. Briefings in Bioinformatics 9: 286–298.

Lassmann, T. 2020. Kalign 3: multiple sequence alignment of large datasets. Bioinformatics 36: 1928–1929.

Lassmann, T. & E. L. Sonnhammer. 2005. Kalign—an accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics 6: 298.

Due at the start of class March 21.