Phylogeny Course Manual
I. From Sequences To Reliable Alignments

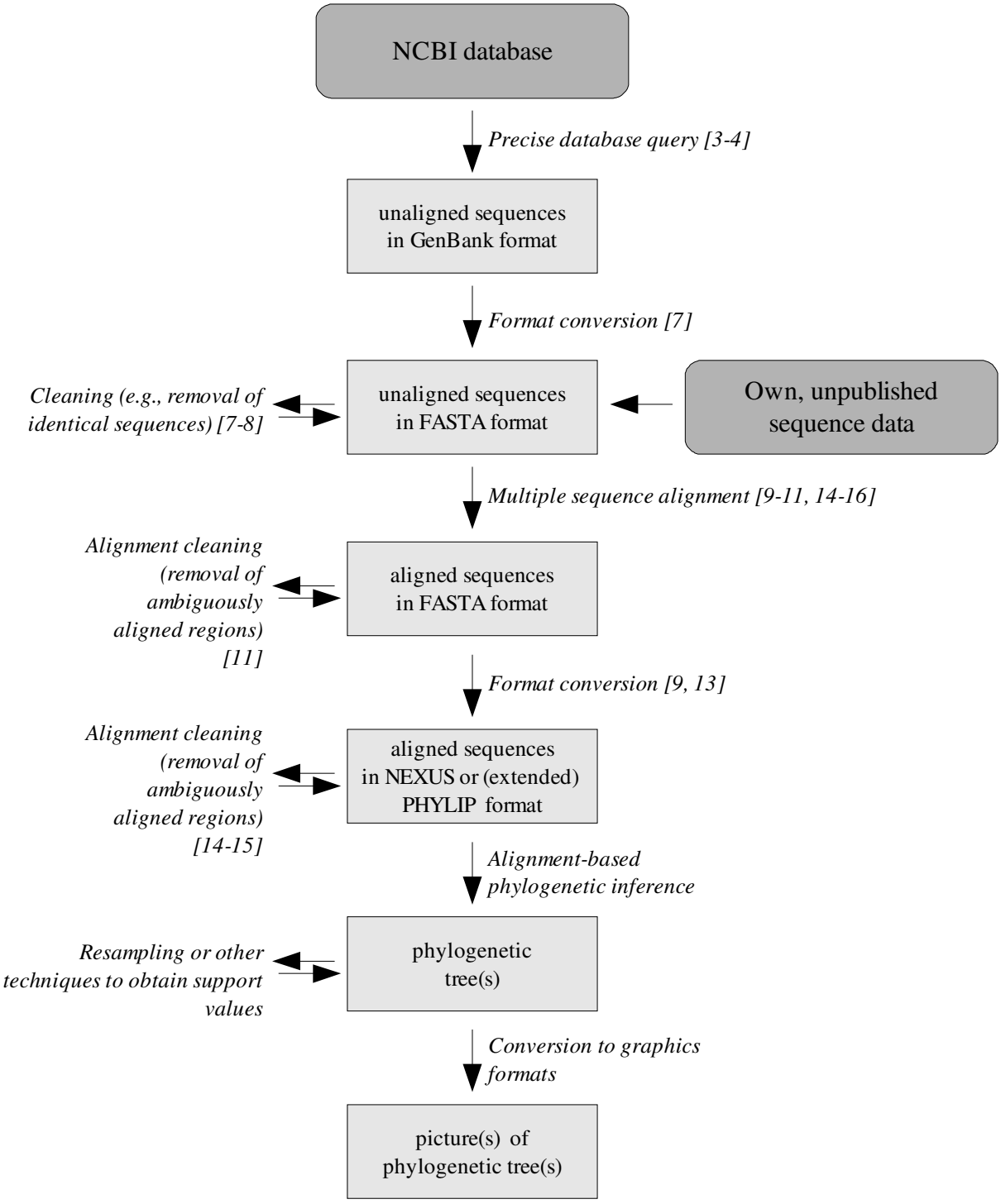
by Markus Göker, Tübingen; version of 04/03/08

These instructions have been compiled for the phylogeny course organized by Michael Weiß of the University of Tübingen Organismic Botany/Mycology department in March 2007. They by no means represent a comprehensive introduction to phylogenetic techniques. Here, I focus on obtaining sequences from GenBank, conversion between formats, state-of-the-art alignment techniques, and removal of ambiguously aligned columns in an objective and reproducible manner. Considerable emphasis is placed on the use of the UNIX command-line.

This document and its accompanying shell scripts are available online at http://www.goeker.org/mg/course/. If you want to use it, please watch the home page for corrections, additions, and other sorts of updates. The author may by contacted at markus [dot] goeker [at] uni [hyphen] tuebingen [dot] de (note that I always respond to e-mail messages, except I fail to do so).

Overview: From sequence to phylogeny

The picture below provides an overview of the steps and techniques necessary to obtain a molecular phylogenetic result to be published. The present manual treats all of these steps except the last two. Numbers in brackets indicate page numbers.



Structure of a GenBank flatfile

When conducting searches at the NCBI web server (http://www.ncbi.nlm.nih.gov/), it is important to know about the features stores in GenBank entries that can be searched for and which appear in the plain text files after download. The following example highlights the most important entries:

```
LOCUS       AY239130                 246 bp    DNA     linear   VRT 30-JUN-2003
DEFINITION  Alligator sinensis 12S ribosomal RNA gene, partial sequence;
            mitochondrial gene for mitochondrial product.
ACCESSION   AY239130
VERSION     AY239130.1  GI:31322847
KEYWORDS    .
SOURCE      mitochondrion Alligator sinensis (Chinese alligator)
  ORGANISM  Alligator sinensis
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Archosauria; Crocodylidae; Alligatorinae; Alligator.
REFERENCE   1  (bases 1 to 246)
  AUTHORS   Gatesy,J., Amato,G., Norell,M., DeSalle,R. and Hayashi,C.
  TITLE     Combined support for wholesale taxic atavism in gavialine
            crocodylians
  JOURNAL   Syst. Biol. 52 (3), 403-422 (2003)
   PUBMED   12775528
REFERENCE   2  (bases 1 to 246)
  AUTHORS   Gatesy,J.
  TITLE     Direct Submission
  JOURNAL   Submitted (16-FEB-2003) Biology, University of California,
            Riverside, Riverside, CA 92521, USA
FEATURES             Location/Qualifiers
     source          1..246
                     /organism="Alligator sinensis"
                     /organelle="mitochondrion"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:38654"
     rRNA            <1..>246
                     /product="12S ribosomal RNA"
ORIGIN
        1 gacttgacgg cgcttcgaac ccacctagag gagcctgtcc tataatcgac ggtacacgat
       61 tcacccgacc acctctagcc cctcagcctg tataccgccg tcgccaagcc cgtccccctg
      121 agggagacaa aacgagcaca atagcctccc aggctagcac gtcaggtcaa ggtgcagcca
      181 atgaggtgga agagatgagc tacattttct aacacataga aatatgcaac ggagagccct
      241 gtgaaa
//
```

These are, in order: the title of the sequence (light grey background); its accession ID (dark grey background), the organism's valid name (bold face); the organism's complete taxonomy as stored in the NCBI taxonomy database (light grey background); the organism's valid name (bold face); the organism's taxon ID as stored in the NCBI taxonomy database (dark grey background); and the sequence (light grey background).

We would retrieve this sequence by searching for, e.g.

"Alligator sinensis"[organism] AND "12S ribosomal RNA"[title]

The quotes are necessary to request the correct word order (i.e., "Alligator sinensis" but not just "sinensis" and "Alligator" anywhere in the organism entry). The names enclosed in brackets specify certain GenBank

fields. Boolean operators such as "AND" or "OR" are available for combined searches. As usual, "OR" has lower precedence than "AND", but parentheses can be used to override the precedence rules. For instance, the following search string:

Asteraceae[organism] AND ("Internal transcribed spacer 1"[title] OR "ITS1"[title]) AND ("internal transcribed spacer 2"[title] OR "ITS2"[title]) AND 5.8S[title]

works well in receiving all sequences of Asteraceae containing the complete ITS region.

Exercises

- Make sure you understand how the example ITS search request works. Assemble search strings for other loci and other taxonomic groups, and try them with the NCBI web server.

Using the UNIX command-line: part I

Terminal programs and shells

Several programs to emulate a UNIX terminal are available for MacOSX. We recommend X11 (if it is installed) since X11 graphical applications such as NEdit (http://www.nedit.org/) can directly be launched from its command-line. On Windows systems, UNIX tools can be used if Linux/UNIX emulators such as Cygwin (available from http://www.cygwin.com/) are installed. Note that the scripts available at our web site (http://www.goeker.org/mg/course/) and used in this manual require the installation of bash, mawk or gawk, and GNU sed. Installation on MacOSX is accomplished conveniently with Fink (http://www.finkproject.org/). The setup.exe file for Cygwin can also be used to install or update single programs after the main Cygwin installation. On Linux/UNIX systems, you mostly can use the native package management system.

Usually, at least two different shells are installed on UNIX-like computers (a shell is the program which interprets the command texts entered at the terminal), the Tenex C shell (tcsh) and the GNU bourne-again shell (bash). The following instructions refer to the latter, but many of them equally apply to the tcsh.

User-specific bash configuration is commonly stored in the files .bash_profile, .bashrc, and .bash_aliases in the respective user's home directory. (The home directory is the directory directly above the Desktop or Documents folders). These are (adapted to MacOSX) provided at the beginning of the course and at our web site. You have to unpack the file bash.zip directly in your home directory; the archive can be removed after unpacking.

Typing conventions

In the following instructions, commands to be entered at the UNIX terminal are typed in monospace font and start with a dollar sign ($). Within these commands, placeholders for file names, command options, etc. are printed in italics, e.g.:

```
$ cat file1 >> file2
```

(This example shows how the cat utility can be used to append the contents of a file to another file, as described in greater detail below.)

General remarks on command-line tools

UNIX commands entered on the command line can usually be invoked with specific command **options**. Mostly, options are either short options starting with a hyphen and consisting of a single character only, e.g.

```
$ command -a -b -c file1 file2
```

or represent long options starting with two hyphens and consisting of a complete word, e.g.

```
$ command --option1 --option2 file1 file2 file3
```

Some programs use both types of options, others use long options that start with a single hyphen only. Often, some of the options have arguments, e.g.

```
$ command --input file1 -z somezargument --hurryup --noerrorplease
```

This implies that in many cases you can get a useful help message by typing either "program -h" or "program --help" or "program -help".

A couple of **keyboard signals** may be used to modify the run of a program. If a program "hangs" and doesn't create any output for a while, it may be that it is just waiting for input. In that case, type Control+D (hold the Control button and type D). A process started on the command line may also have become stuck because of a programming error (a "bug", the programmer's fault) or erroneous input (your fault). In that case, enter Control+C.

Tab completion is a shell feature that saves much typing. Once you have entered the beginning of a word, pressing the tab button on the keyboard results in an attempt by the shell to complete the word. The shell will stop at the position after which a unique expansion is impossible. For instance,

```
$ gbk2[TAB]
```

will expand to

```
$ gbk2fas.sed
```

(the name of a program explained below). Note that word expansion can be used for command names (at the beginning of a line) as well as for file or directory names.

The shell's **history** mechanism is also an extremely useful feature. During an interactive shell session, bash will store all commands entered successfully (i.e., not resulting in an error message) in a history file. You can use the arrow-up and arrow-down buttons to browse within you history. In that way, you can easily re-enter commands typed earlier, with or without modification.

File and directory management commands

```
$ cp file1 file2
```

Places a copy of file1 in file2. Directories can be copied with the same syntax.

```
$ cp file1 dir1
```

Puts a copy of file1 in directory dir1; the file name remains "file1". Several file names can be provided.

```
$ mv badfile goodfile
```

Renames badfile to goodfile. Directories can be renamed with the same syntax.

```
$ mv file1 dir1
```

Moves file1 in directory dir1; the file name remains "file1". Several file names can be provided.

```
$ rm badfile
```

Deletes badfile. Several file names can be provided.

```
$ rm *.txt
```

Removes all files in the current working directory which end in ".txt". The star "*" is a so-called wildcard expression, implying "zero or more arbitrary characters".

```
$ rm –r ~/*
```

Removes everything in your home directory (you probably don't want to do that). The tilde "~" is a placeholder for your home directory. "-r" tells the rm program to recursively enter all subdirectories. The star again acts as a wildcard, ensuring that everything is deleted.

```
$ pwd
```

Print the name of the current working directory.

```
$ mkdir ~/Desktop/dir1
```

Creates the directory dir1 on your Desktop.

```
$ cd ~/Desktop/dir1
```

Use the directory dir1 on your Desktop as the working directory (i.e., move into that directory).

```
$ cp ..
```

Use the parent directory of the current one as working directory.

```
$ cd –
```

Use the directory where you have been before as working directory.

```
$ cd
```

Move into your home directory.

```
$ rmdir dir1
```

Remove directory dir1. This only works with empty directories; use "rm -r" otherwise.

```
$ ls ~/Desktop
```

List the contents of your Desktop folder.

```
$ ls –al ~/Desktop
```

List the contents of your Desktop folder in tabular format (-l) including various file and directory information and also display all hidden files the names of which start with a dot (-a).

Standard input/output/error and redirection

By default, UNIX uses the terminal as standard input (stdin), standard output (stdout), and standard error (stderr). That is, an application reading from standard input expects something typed on the terminal; a program putting its results to standard output will print it on the screen; and in case of an error occurring during execution, a utility will usually display its error messages on the screen. However, all three special file descriptors can be redirected, as shown with the cat program:

```
$ cat file1
```

Print the contents of file1 on the screen (cat interprets its command-line argument as a file name).

```
$ cat < file1
```

Also print the contents of file1 on the screen (cat uses the contents of file1 as standard input).

```
$ cat file1 > file2
```

Overwrite file2 with the contents of file1 or create file2 if it doesn't exist. Here, standard output is redirected to file2. (Standard error is still printed to the screen!)

```
$ cat file1 >> file2
```

Append the contents of file1 to file2. Again, file2 will be created if it doesn't already exist and standard error is still printed to the screen.

```
$ cat file1 2> file2
```

Write any error messages occurring when applying cat to file1 into file2, replacing any previous contents. If no error occurs, an empty file file2 is created.

```
$ cat file1 2>> file2
```

Append any error messages occurring when applying cat to file1 into file2. If no error occurs, an empty file file2 is created if it doesn't already exist.

Reading text files

The cat utility is inconvenient to read large plain text files because everything not fitting on the screen is lost. "Pager" programs such as the "less" utility display each page of a file separately:

```
$ less file1
```

You can quit the program by typing "q" and move around in the document with arrow keys. A complete list of less commands is displayed if you type "h".

Many text files won't be displayed nicely with less because they don't use the correct (UNIX) line break characters. Unfortunately, native Macintosh application still use the carriage return (CR) as the character separating lines in text files instead of the line feed (LF) used by UNIX systems (DOS/Windows uses the combination, CR+LF, which is less error-prone than CR). **Note that non-UNIX style line termination frequently leads to severe errors in sequence file format conversions.** Probably the simplest command to convert your files to UNIX line breaks is

```
$ strings file1 > file2
```

You can also apply

```
$ tr '\r' '\n' < file1 > file2
```

which is safer, but somewhat more inconvenient to type.

Converting GenBank to FASTA format, and a simple cleaning procedure

As a first "real" command-line application, we convert sequence files downloaded from GenBank (http://ncbi.nlm.nih.gov/). Having downloaded your files in GenBank flatfile format, you commonly want to convert them to FASTA format (e.g., as input for alignment programs), including only the most important information in the FASTA headers. The tool gbk2fas.sed (available at http://www.goeker.org/mg/course/) assumes that you conduct an organismic study and that you want to keep the organism's taxonomically valid name and the sequence accession number in the FASTA definition line. Just type

```
$ gbk2fas.sed genbankfile > fastafile
```

In contrast, we do not recommend downloading files in GenBank's own FASTA format since its definition

lines do not necessarily contain the proper taxon name.

For training purposes and in a "real" phylogenetic investigation that deals with relationships between taxa of higher rank, it is usually inconvenient if datasets contain too many identical sequences. We provide a simple utility that removes duplicate sequences from FASTA files. Sequences which represent substrings of (i.e., are entirely contained in) other sequences are also removed; from each group of identical sequences, one of the longest sequences is output. The script is invoked as follows:

```
$ dufa.awk –v L=sequences2.log sequences1.fas > sequences2.fas
```

This will produce a mostly somewhat smaller FASTA file (here, "sequences2.fas") and a log file (here, "sequences2.log") providing information about the groups of identical sequences recognized and about which sequence from each group was kept. Information on total dataset reduction is printed to standard error. Note that an important limitation of our script is that it does not recognize nucleotide (or amino acid) ambiguity code.

References

- The Linux documentation project (http://www.tldp.org/guides.html) covers the standard command-line tools mentioned above and many more.

Exercises

- Create, rename and remove directories, copy and rename files, copy and place them in different folders.
- Try to remove files or directories which do not exist and watch the error messages. Try to move, copy, or read files which do not exist and watch the error messages.
- Concatenate the contents of several files into a single large file.
- Create text files using a native Macintosh text editor, try to find out whether they have UNIX line terminations and try to convert them in case they do not. Try to display binary files with less.
- Do all these things with the files you have downloaded from GenBank. Convert them to FASTA format and check whether the definition lines look as described above. Use only command-line tools to accomplish these tasks.

EMBOSS utilities

EMBOSS is the "The European Molecular Biology Open Software Suite".

Obtaining EMBOSS

The EMBOSS source code can be downloaded for free at http://emboss.sourceforge.net/. Installation instructions are also provided at the EMBOSS website. A graphical user interface for EMBOSS, JEMBOSS, can also be downloaded there.

General remarks

The EMBOSS package contains a large number of specialized tools. During the course, we will make use of only a few of them. A complete list of EMBOSS utilities can be printed using

```
$ wossname -auto
```

wossname can also conduct searches in the documentation of all EMBOSS programs (omit "-auto" and enter your search string at the wossname prompt). The help message for each of the programs can be obtained by typing

```
$ program -help -verbose
```

at the UNIX prompt. An even more verbose help is provided by

```
$ tfm program
```

where "program" is the name of the EMBOSS utility of interest.

seqret

seqret is a very useful tool to convert between different sequence formats. Both aligned and unaligned sequences are supported. Usage is:

```
$ seqret -auto -sequence input_file -outseq output_file -osformat2 output_format
```

Most input file formats are recognized automatically; an input format can be explicitly provided using the -sformat1 switch. Default output format is FASTA. For a complete list of supported formats, see the http://emboss.sourceforge.net/docs/themes/SequenceFormats.html web page. To obtain non-interleaved NEXUS format, use -osformat2=nexusnon (interleaved NEXUS format is not recommended since this may lead to truncation of sequence labels). However, you can also convert to PIR format with seqret and convert the PIR format to NEXUS with PAUP* by typing

```
paup> tonexus format=pir fromfile=sequences.pir tofile=sequences.nex;
```

at the PAUP* prompt (symbolised by “paup>”). This is safer because PAUP* will adjust the sequences labels to valid NEXUS format. Note that seqret does not examine whether the sequences are aligned; converting unaligned sequences to NEXUS or PHYLIP format will result in syntactically invalid files.

degapseq

This works like seqret, but removes all gap characters from aligned sequences. Typical usage is as above, just replace "seqret" by "degapseq".

seqretsplit

It is sometimes useful to put each sequence in a single file, a task seqretsplit has been written for:

```
$ seqretsplit -auto -sequence input_file
```

File names are created from the FASTA headers. Unfortunately, they are written in lower case letters only.

infoseq

infoseq is a simple utility to display information (name, length, GC content) about the sequences contained in a file:

```
$ infoseq input_file
```

transeq and tranalign

Whereas transeq can translate nucleotide to amino acid sequences using the specified reading frame, tranalign will align nucleotide sequences provided an alignment of their corresponding amino acid translations. This will usually result in greater accuracy than the proper nucleotide alignment (as a more elegant approach, implicit translation is provided by Dialign, see below). transeq can be invoked like this:

```
$ transeq -auto -clean -frame number -sequence nucleotide_sequences -outseq peptide_sequences
```

where "number" is one of 1, 2, or 3. "-clean" changes stop codon ("*") to missing ("X") characters since the former are not recognized by some alignment programs. Additionally, "-osformat2" can be used to choose output sequence format. Usage of tranalign is:

```
$ tranalign -auto -asequence unaligned_nucleotide -bsequence aligned_protein -outseq aligned_nucleotide
```

Note that the sequences have to be in the same order in both input files!

References

- Rice, P., Longden, I., and Bleasby, A. 2000. EMBOSS: The European Molecular Biology Open Software Suite. Trends in Genetics 16(6): 276-277.

Exercises

- Use seqret to convert back and forth between FASTA, PIR, PHYLIP, and NEXUS sequence format. Create both NEXUS and PHYLIP interleaved and non-interleaved formats and read the resulting files.
- Explain the differences between these formats; have a look at the example files at http://www.goeker.org/mg/course/sequence_formats.zip.
- Which of the formats can hold both aligned and unaligned data? Which of them are only useful to hold aligned data?
- Search additional interesting EMBOSS programs, read their help and apply them to your sequence data.

Multiple sequence alignment: part I

MAFFT and MUSCLE

Both MAFFT and MUSCLE are multiple sequence alignment (MSA) programs still under active development. The following instructions refer to MAFFT version 5.861 (http://align.bmr.kyushu-u.ac.jp/mafft/software/) and MUSCLE version 3.6 (http://www.drive5.com/muscle/). Both programs use a strategy of iterative improvement and have been optimized for aligning amino acid sequences, but work very well with nucleotides, too. A couple of command-line options is available which fine-tune the programs' behaviour; as usual, there is a trade-off between speed and accuracy of the results. We focus on options that have worked reasonably well with DNA and finish in time.

MAFFT may be used both interactively and non-interactively. Interactive use consists of typing the

application's name and answering the questions it prints to the screen. Of course, non-interactive use is easier to standardise:

```
$ mafft --quiet --maxiterate 1000 unaligned.fas > aligned.fas
```

For the fastest possible run, use the same commands as above without "--maxiterate 1000".

We recommend to start MUSCLE as follows:

```
$ muscle -quiet -stable -in unaligned.fas -out aligned.fas
```

If you wish to observe the alignment progress, omit "--quiet" or "-quiet", respectively. "-stable" ensures that the order of sequences is the same as in the input file. This is the default behaviour of MAFFT but not of MUSCLE and is strongly recommended. Alternatively, the sequences are output in the order they've been aligned. The fastest possible invocation of MUSCLE (with nucleotide sequences) is:

```
$ muscle -maxiters 1 -diags -quiet -stable -in unaligned.fas -out aligned.fas
```

Muscle may also be used to combine pre-existing alignments ("profile alignments"). For that purpose, use

```
$ muscle -profile -stable -in1 unaligned1.fas -in2 unaligned2.fas -out aligned.fas
```

This can be convenient if you have to combine partially overlapping datasets (such as, e.g., ITS & LSU rDNA sequences as well as ITS-only sequences).

Gblocks

Our Gblocks description is based on version 0.91b. Gblocks (http://molevol.ibmb.csic.es/Gblocks/Gblocks.html) is a simple utility to remove the least conserved (and possibly ambiguously aligned) columns from an alignment. For instance,

```
$ Gblocks aligned.fas -t=d -b5=h
```

will regard the sequences as DNA and will include columns with gaps in up to the half of the sequences. Alternatively, "-t=p" forces treatment as protein sequences and "-t=c" as codons. "-b5=n" can be used to remove all gapped columns and "-b5=a" to retain all of them. A reduced FASTA file called "aligned.fas-gb" and a HTML document "aligned.fas-gb.htm" would be created from "aligned.fas" input. The colourful HTML file shows the columns selected for alignment. Use the gblocks2nexus.sh script provided at http://www.goeker.org/mg/course/ to directly obtain NEXUS files using Gblocks:

```
$ gblocks2nexus.sh aligned.fas > aligned.nex
```

Seaview

Seaview is an alignment viewer and converter freely available for all common operating systems. It can be obtained from the http://pbil.univ-lyon1.fr/software/seaview.html web site. It supports several formats, including FASTA, PHYLIP, and NEXUS, and can thus be used for format conversions. Individual sequences and gap-only columns can be removed with Seaview, too.

References

- Castresana, J. 2000. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Molecular Biology and Evolution 17: 540-552.

- Edgar, R. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput Nucleic Acids Research 32: 1792-179.
- Galtier, N., Gouy, M, and Gautier, C. 1996. SEAVIEW and PHYLO_WIN: two graphic tools for sequence alignment and molecular phylogeny. Comput. Appl. Biosci. 12: 543-548.
- Katoh, K., Misawa, K., Kuma, K., and Miyata, T. 2002. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Research 30: 3059-3066.
- Katoh, K., Kuma, K., Toh, H., and Miyata, T. 2005. MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucleic Acids Research 33: 511-518.

Exercises

- Align the same sequences with both MAFFT and MUSCLE and compare the results (use the SEAVIEW application to watch the aligned sequences; note that it does not display several alignments simultaneously).
- Read the help of MAFFT and MUSCLE and try to find out what the other options do.
- Why may the order in which the sequences have been aligned be different from the input order?
- Align datasets containing different numbers of taxa and sequences of different mean length and observe the differences in the alignment utilities' running times.
- Use Gblocks to remove alignment columns. Observe how many and which columns are deleted.
- Why is it "simple" to regard alignment columns with many gaps as ambiguously aligned?
- Use seqret to further process the aligned data.

Using the UNIX command line: part II

A few remarks on process management

Computations in phylogenetic inference may consume a large amount of both time (running time) and space (computer memory). It is therefore convenient to know how to observe resource consumption of running processes and to change it if necessary. The command

```
$ top
```

Provides a list of those processes which need most CPU power and memory. Total running time, the processes' owner (the user who started the process) and the process identification number (PID) are also shown. The PID can be used to terminate a process if you are its owner or a privileged user:

```
$ kill PID
```

Several options to kill are present, indicating how violently the process should be aborted (e.g., whether the process gets a chance to clean up memory before stopping). If you know the name of the process, you can also use

```
$ killall name_of_process
```

but keep in mind that this will really attempt to kill all processes with that name.

Often it is inconvenient to start a process from a terminal in the usual manner because you want to type additional commands. In that case, start the process in the background by typing a single ampersand "&" after the program invocation, e.g.

```
$ muscle –in unaligned.fas –out aligned.fas –stable –quiet &
```

The shell will provide you with the process number and later on also with a message that the process was finished. Note, however, that starting an interactive process (e.g., a PAUP* session) in the background doesn't make sense and sometimes leads to an immediate error. Further, standard output of a background process will still appear on the screen unless you take special care (e.g., the "-quiet" option in the example above).

Processes can have different priorities; those with highest priority (usually -20; 19 is lowest) get more resources. You can start a process with an explicit priority, e.g.

```
$ nice 5 muscle –in unaligned.fas –out aligned.fas –stable –quiet &
```

(If you set a low priority, you are "nice" to other users.) Note that on some systems you would have to type "nice -n 5 muscle..." and that nice is a built-in command of the tcsh, but also a program usually located at /usr/bin/nice. Priority of an already running process can be altered using the command "renice" in combination with the process ID (i.e., if you feel unobserved you can start being less nice than before). On a single-user machine, these commands may be useful if you start a long phylogenetic analysis but also want to do some other work with your computer.

Getting scripts and programs executable

If you do not have administrator privileges, it is convenient to create a folder for executables in your home directory:

```
$ mkdir –p ~/bin
```

In order to start the scripts or programs located in that folder just by typing their name, it is necessary to include its name in your PATH variable. This can be done automatically at startup; see the ".bash_profile" file in the bash.zip archive at http://www.goeker.org/mg/course/ for an example. After creating the bin folder, place bash.zip in you home folder and enter:

```
$ unzip –o bash.zip
$ rm –f bash.zip
```

Afterwards, you should log out and open a new shell window, or alternatively enter:

```
$ source ~/.bash_profile
```

To make a file executable, you have to change its access permissions. The following command ensures that every user of the system can read and execute the file "example.exec", but only its owner (or a privileged user) can delete it:

```
$ chmod 755 example.exec
```

If the file should not be executable, enter "644" instead of "755"; if other users should not be allowed to read it, use "600". Executable scripts usually represent code in a text file which has to be read by an interpreter the location of which usually is provided in the first line of the code, e.g.

```
#!/bin/sed –f
```

This will not result on MacOSX systems because if GNU sed is installed at all it is located at /sw/bin/sed (at least if you use fink for installation) and another sed is found at /usr/bin/sed, but none in /bin/sed. You can try to find out where the interpreter is by typing

```
$ which name_of_the_interpreter
```

and change the first line of the script code accordingly. Afterwards, place the executable in the ~/bin directory. The program's name should be immediately visible if ~/bin is contained in the PATH variable.

Converting to extended PHYLIP format

This topic does not really fit to any of the other headlines, so I have tried to hide it here. Some phylogenetic inference packages such as PHYML, RAxML or PAML use a sequence format which unfortunately is not covered by the EMBOSS package. It is a modification of the well-known PHYLIP format and mostly called "extended PHYLIP". We provide a simple-minded converter from NEXUS to extended PHYLIP, invoked like this:

```
$ nex2epf.sed aligned.nex > aligned.epf
```

Keep in mind that this converter does not really "understand" NEXUS format and may fail even if the file is valid NEXUS. It will work, however, with non-interleaved NEXUS format as exported by PAUP* if you enter:

```
paup> export format=nexus interleaved=no linebreaks=unix file=aligned.nex;
```

at the PAUP* prompt, as long as you do not later on modify the data block in an unusual manner.

Exercises

- Start a time-consuming process (e.g., an alignment of a large numbers of sequences) in the background. Observe it using "top" and stop the process with "kill".
- Start several CPU-intensive processes and observe them with "top". Kill these processes and start them again using "nice" and different priorities. Observe the differences with "top" before and after applying "renice".
- Use nex2epf.sed to convert NEXUS to extended PHYLIP and seqret to convert to PHYLIP. Explain the differences between PHYLIP and extended PHYLIP.

Multiple sequence alignment: part II

Dialign

The Dialign multiple sequence alignment program is unique in that it does not rely on gap insertion and expansion penalties but constructs the global alignment from sequence fragments collected and associated with a certain local score. These segments are successively put together in a consistent manner (i.e., in agreement with co-linearity). As a consequence, Dialign (we are referring to version 2.2.1; see http://bibiserv.techfak.uni-bielefeld.de/dialign/) works particularly well if the sequences are only locally related. An interesting options is that nucleotides may implicitly be translated to amino acids, enabling homology assessment at the peptide level, too. Therefore, Dialign is pretty good in aligning coding sequences with interspersed introns.

Importantly, some segments within the sequences may be left and are not considered to be aligned. These fragments are printed in lower case and must be coded as "missing data" in phylogenetic analysis. Dialign also outputs scores from 0-9 related to the alignment column reliability. These can be used to recognize ambiguously aligned regions and to exclude them before searching for trees (Kemler et al. 2006). We provide

a shell wrapper script dia2nex.sh (http://www.goeker.org/mg/course/) which runs Dialign and translates its output to a NEXUS file with appropriate charset commands for each score value.

Synopsis: common Dialign and dia2nex.sh options		
dia2nex.sh	**Dialign**	**implication**
-f	-fa	output a FASTA file, too
-m	-n -ma	assume nucleotide input, but assess homology also at the amino acid level
-p	[default]	assume amino acid input
-s	-o	slightly speed up alignment by using approximations
-t	-nt	assume nucleotide input, but assess homology at the amino acid level only

I.e., in contrast to Dialign, default dia2nex.sh sequence input is nucleotide. Typical usage is:

```
$ dia2nex.sh unaligned.fas > aligned.nex
```

which is (except naming conventions) equivalent to directly starting Dialign with

```
$ dialign2-2 -n unaligned.fas
```

Note that dia2nex.sh can also convert a pre-existing Dialign .ali file to NEXUS format. For further options, please read the command-line help. After executing the NEXUS file in PAUP*, you can exclude the least reliably aligned columns by typing

```
$ exclude score_0 score_1;
```

The Dialign scores are also converted to weights, but their use is not recommended.

POA

Version 2 of the POA multiple sequence alignment program is available from the http://sourceforge.net/project/showfiles.php?group_id=168080 web site. POA uses so-called partial order graphs to represent MSAs. This data type avoids averaging of fitting scores if sequences are to be added to a growing MSA. Accordingly, even though POA does not use iterative refinement but in default mode just aligns sequences in input order, accuracy of the results is comparable to the alignment programs mentioned earlier. POA is relatively fast and has a low memory consumption even with large numbers of sequences, particularly in progressive mode, which is invoked by supplying pair-wise sequences similarity scores. POA is also good in aligning sequences with very long indels.

Command-line use of POA is a little bit complicated; therefore, we have created a shell wrapper aop.sh (available from http://www.goeker.org/mg/course/). The wrapper has some additional features such as providing the scoring matrix for POA, pre-cleaning FASTA files, and computing pair-wise similarity scores with different algorithms to start POA in progressive mode. Regarding the latter, a useful feature is to extract pair-wise taxonomical similarity scores from GenBank flatfile. That is, sequences are aligned in the order of the taxa's phylogenetic relationships according to the current NCBI taxonomy.

Synopsis: common POA and aop.sh options		
aop.sh	**POA**	**implication**
-a	-read_msa	read multiple sequence alignment file (the unaligned sequences are progressively aligned with) from given file
-f	-remove	read names of sequences to be removed from alignment input from given file (the aop.sh -f option also affects GenBank flatfile input)
-p	[default]	assume amino acid input
-q	-printmatrix	print scoring matrix to standard output
-s	-read_pairscores	read pair-wise similarity scores from given file

I.e., in contrast to POA, default aop.sh sequence input is nucleotide. Typical usage is:.

```
$ aop.sh –c sequences.fas
```

which will create the files sequences_poa.fas (Alignment in FASTA format), sequences_poa.pog (Alignment in partial order graph format), and sequences_poa.out (a logfile showing the alignment steps). This is equivalent to

```
$ poa –read_fasta sequences.fas –preserve_seqorder –tolower –v –pir sequences_poa.fas –po sequences_poa.pog Blosum_matrix
```

except that POA uses the dot as gap symbol and may truncate sequence labels. "Blosum_matrix" is the external scoring matrix. We have extended the matrix distributed with POA by adding nucleotide ambiguity code symbols (the extended matrix is available separately at http://www.goeker.org/mg/course).

The -a (-read_msa) option is useful for working with profile alignments (see description of MUSCLE above). For further options of aop.sh, please read the command-line help. Post-processing of POA alignments can be done with the tools described earlier (seqret, Gblocks, etc.).

References

- Grasso, C. and Lee, C. 2004. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. Bioinformatics 20: 1546-1556.
- Kemler, M., Göker, M., and Begerow, D. 2006. Implications of molecular characters for the phylogeny of the Microbotryaceae (Basidiomycota: Urediniomycetes). BMC Evolutionary Biology 6:35 [Introducing the use of Dialign scores to exclude alignment columns].
- Lee, C., Grasso, C., and Sharlow, M. 2002. Multiple sequence alignment using partial order graphs Bioinformatics 18: 452-464.
- Morgenstern, B. 1999. DIALIGN2: improvement of the segment-to-segment-approach to multiple sequence alignment. Bioinformatics 15: 211-218.
- Morgenstern, B., Frech, K., Dress, A, and Werner, T. 1998. DIALIGN: Finding local similarities by multiple sequence alignment. Bioinformatics 14: 290-294.

Exercises

- Compare the results from Dialign and POA alignment with each other and with those obtained with MAFFT or MUSCLE.
- Apply all four programs to datasets with large numbers of sequences and compare the running times.
- Compare the results of Dialign and exclusion of the low-scoring columns with those obtained with alignment with any of the three other programs (MAFFT, POA, and MUSCLE) combined with Gblocks exclusion.
- If you have coding nucleotide sequences, translate them using tranalign, crate an amino acid alignment with any of out four alignment programs, and translate the peptide alignment back to nucleotide using tranalign. Compare the results to those obtained by directly aligning the nucleotide sequences.
- Does an externally provided alignment guide tree (or distance/similarity scores representing such a tree) induce a bias in subsequent phylogenetic analyses? If so, which kind of bias?

Using the UNIX command line: part III

Compiling programs

As an example for the process of compiling programs (i.e., roughly spoken, of translating source code to executables that run on your system), we consider the POA alignment program. Download the file poaV2.tar.gz from the web site (see above) and place it, e.g. on the desktop. Change your working directory to the desktop and enter:

```
$ tar –xzf poaV2.tar.gz
```

to unpack the archive. A folder poaV2 is created. Move into that folder and read the README file. Compiling the programs is obviously simple, just enter

```
$ make poa
```

In order to compile the programs written in C, you need a C compiler and almost always also the Make utility. Most programs on Linux/UNIX systems are developed with the GNU compiler collection (gcc). For programs written in C++ such as MUSCLE, you can use the g++ compiler; for those in Fortran, g77; for those in Ada, Gnat. See above for instructions on installing such packages. The name(s) of the executable(s) resulting from successful compilation are also stated in the README file. As usual, you can place them in your ~/bin directory.

Exercises

- Try compiling the other alignment programs mentioned in this manual. Search for the source code archives on the web sites, download and unpack them. Read the README or INSTALL files or any other documentation carefully and follow the instructions.
- Download pre-compiled executables for the same programs from the web sites (if available) and compare them to the results of you own compilations.
- If compilation doesn't work, try to find out which necessary package is missing (Make, gcc, etc.).