Lecture 3: Bio599

Greg Caporaso
gregcaporaso@gmail.com

Question from last time: why lowercase letters in BLAST output?

- BLAST treated these characters a region of "low complexity sequence". These are often things like repeat regions that can cause spurious hits in BLAST results.
- See DustMasker (MORGULIS, 2006; Journal of Computational Biology)

Regular expressions

- Language for search/replace
- Widely used, but unfortunately not completely standardized.
  - grep, python, TextWrangler, jEdit, perl, java, and the list goes on…
  - Each may have small idiosyncrasies, so to get started choose your weapon and stick with it.
  - Google for your tool and “regular expression” will usually turn up useful reference pages

Reformatting text to move from one program to another

- Replace commas with tabs in delimited text
- Sometimes too complex for basic search/ replace:
  - search term is too generic
  - input is too varied to match with a simple term
  - columns need to be re-ordered
  - information needs to be moved from one entry to another

Manual manipulation of files is tedious, error-prone, and a huge waste of time!!

Regular expressions

- More powerful approach for search and replace.
- Easy to perform basic functionality (e.g., replace *human* with *Homo sapiens*)
- Employ *wildcards* to match varied patterns (e.g., all digits, but where you don't know exactly what digits you're expecting).
- Capture parts of a search term and use it in the replace term.

Wildcards

- `\w` : matches any single digit or letter

  A-Z, a-z, 0-9

Appendix 2 of PCFB is your reference.

Regular expressions are case sensitive

- `\w` is not the same as `\W`
- searching for `Agalma` is not the same as searching for `agalma`

Capturing text

- To *capture* text for later use, you can use `( )`
- You can then reference that text with `\1  \2` and so on… (In jedit, you'd use `$1  $2` and so on…)
- Search and replace terms can contain normal text and regular expressions.

Quantifiers

- Specify how many times a pattern should occur to be matched.
  - + : match one or more occurrences
  - * : match zero or more occurrences
  - ? : match zero or one occurrences

Escaping special characters

- With characters like + and (having special meanings, how do you match these characters in text?
- To remove the special meaning, precede the character with a \

Wildcards

`\w` : any single digit or letter

`\t` : tab character

`\s` : any white space character

`\r` `\n` : new line characters (can be different in different tools)

`\d` : a digit (0-9)

`.` : any letter, number or symbol except \r or \n

Appendix 2 of PCFB is your reference.

Steps for building a regular expression

1. Copy the text you want to match to a new file
2. Mark the areas you'd like to capture
3. Add wildcards (maybe include spaces, if that helps)
4. Remove any extra spaces that you added.
5. Define the replacement string.

In class exercise 0

- Reformat taxa names to genus abbreviation, species name, name of person who named the species separated by underscores and excluding any parenthesis.

In class exercise 1

- Reformat sequence headers in a fasta file
  - Rewrite each identifier as the portion of the identifier preceding the . character, followed by an underscore, followed by the genus name

In class exercise 2

- Reformat “blast9” output
  - Remove header (i.e. comment) lines
  - Format each line to contain the subject id, the query id, the e-value, the percent identity, and the alignment length, in that order!
  - Format as comma-separated text

Defining custom wildcards

- You'll eventually want to match only certain characters, for example the letters in the nucleic acid or protein alphabet. This is accomplished using the `[ ]` characters:

  `[ACGT]` will match a single A, C, G or T

Wildcard ranges

- `[A-Z]` will match any uppercase letter
- `[a-z]` will match any lowercase letter
- `[A-Za-z]` will match either upper or lower case letters
- `[0-9]` will match any digit

In class exercise 3

- Tab-separated lat/long