Tutorial 05 - Regular expressions and their application

regex clean up

- best practice to put your regular expressions inside quotes (either work)
- in bash, metacharacters (including `\`) are escaped
- `\s` matches any whitespace, except the newline character
- `\t` matches tab and `" "` matches space

grep -E vs. egrep

These are equivalent

You can use either

sed vs. tr

There are a number of things both of these tools can do, but key differences are:

- `tr` works character by character and `sed` works on words (including regular expressions)
- `tr` can find and replace newline characters (`\n`), but `sed` isn't very good at working with those
- if you want to use extended regular expressions with `sed`, don't forget the -E!!!

`tr` application

Challenge: use `tr` and `rev`, to create the reverse complement of the sequence contained in *DNA.txt*

`sed` application

Challenge: use `sed` and a regular expression to replace any four-letter word that starts with ‘b’ with ’book

shell script for finding RNA transcriptase binding sites

You have a series of files with a single DNA sequence in each. You are interested in the N basepair sequence upstream (before) the beginning of a gene (if there is one) in each sequence. Write a shell script that can operate on any user-specified set of DNA sequence files and return a user-specified number of basepairs upstream of a gene from each sequence and store these in a new text file (name specified by the user).

-You can use the sequence files in the zip folder on today's tutorial page on Sakai to test this shell script.

-Each sequence file contains a single sequence, but it is wrapped across multiple lines.

-Remember genes start with ATG and end with TAG, TAA, or TGA. Don't forget about codons!

function review

`grep`

`tr`

`sed`