# Unix, Perl and BioPerl

Session 3: Sequence analysis with Perl (modules including BioPerl)

## Exercise 2: Manipulating a GenBank file with BioPerl and creating a PNG image

Goals: (1) Learn some basic BioPerl commands for handling sequences and sequence features. (2) Learn some basic commands in the GD module and create a PNG image from sequence data.

See `http://jura.wi.mit.edu/bio/education/bioinfo2005/unix-perl/` for course page

Most of this exercise is performed by uncommenting commands. In this manner you can get the feel of the parts of a quite complex script without spending very much debugging time.

**Main functions of the script:**

1. **Parse sequence and feature information from a GenBank report**
2. **Generate a PNG color image from this sequence data**
3. **Print out the gene sequences and the encoded protein in a different format**

- Start the Xwindows system, since you'll need it to view the image files (unless you want to download them first).

- If you haven't already done so, make a directory called perl_2 and copy all files from /home/george/perl_2
  Starting script = `genbank2image.pl`
  data file = `seqs.gb` (a set of mRNA sequences in GenBank format)

- If needed, make `genbank2image.pl` executable.

**The next five bullets help explain the general organization of the script.**

- Look at `seqs.gb` to get an idea of the input data. Find the "FEATURES" and the "ORIGIN" (raw sequence) sections for the first sequence.

- Open the file `genbank2image.pl` The file is complex enough that multiple commands performing related tasks are placed together into a subroutine (ex: `sub readSequence {}`) This subroutine must then be "called" (performed) with the command `readSequence()`.

- Notice the "use" statements (near the top) that calls the BioPerl and GD modules. The key statement for BioPerl is the creation of a SeqIO object with the command like
  ```
  $in = Bio::SeqIO->new();
  ```
  The file format and name must be specified. The "while" loop then creates a sequence object (`$seqobj`) for each sequence.

- With each sequence, the sequence itself is extracted from the GenBank report (in one subroutine). Then the features are parsed (in another subroutine).

- Image creation: An image object ($img) is created for GD. The module isn't smart enough to know usual colors, so any colors must be defined before being used. Also, the `defineNtColors` subroutine links each nucleotide to a color.

**To do:**

- Run the script, ignoring the "used only once" warnings. It should give you a list of png images produced by the script. Open one of these in Netscape (by entering the command "netscape" or selecting it on the main Xwindows menu). It should be mostly blank, with numbers down the left side.

- 1,2: Uncomment these commands to print a filled rectangle and a string (the nt). Run the script again and click "reload" in Netscape.

- 3: Get the numbers indicating the beginning and end of the CDS (CoDing Sequence), make these into a string, adding the string to `$title` (in a meaningful manner) when a CDS is found. If a CDS isn't found, don't add anything to the title.

- 4: (Optional) Go to the `defineColors` and `defineNtColors` subroutines and change the colors for each nt. Each color comes in a (red, green, blue) triplet, and each level is represented from 0 to 255. To get a quick look at potential colors, you can consult a web site like `http://eies.njit.edu/~kevin/rgb.txt.html` or `http://biocomputing5.wi.mit.edu/svg/colors.html`

- 5: As you iterate through each sequence, print a fasta file of the DNA sequence and the protein sequence. Look at the `printSeqs` subroutine, where sequence objects are created for output. Again, one can choose among multiple formats. Uncomment the line where the subroutine is called.

- See /home/george/solutions for a finished script (`genbank2image_SOLUTION.pl`).