Bioinformatics

Computational Methods III:
Sequence Analysis
with Perl and BioPerl

George Bell
WIBR Biocomputing Group

Sequence analysis with Perl and BioPerl

- Regular expressions
- Hashes
- Using modules
- Library for WWW access in Perl (LWP)
- Common Gateway Interface Class (CGI)
- GD graphics library (to generate figures)
- BioPerl (SeqIO, BPlite)

Regular expressions

- "a pattern to be matched against a string"
- found in Unix, Perl, and elsewhere
- used in Perl for matching and substitution
- Regexps use lots of special characters
- Perl example: extracting human fasta deflines

```
@def = grep (/^>.*(human|homo)/i, @lines);
```

`^`	beginning of word anchor
`.`	any character but newline
`*`	0 or more of preceding character
`|`	logical 'OR'
`i`	pattern is case insensitive

Some uses of regular expressions

- biological applications you've seen:
  - protein motifs
  - transcription factor binding sites
- other biological applications:
  - parsing GenBank and BLAST reports
  - reformatting data from a file (ex: EMBOSS output)
  - extracting references from a manuscript

Writing a regular expression

- Describe the pattern in English
- What part of match do you want to extract?
- Translate into Perl (see below)

`[A-Z]`	any capital letter	`\bword\b`	word anchor
`[0-9]*`	>= 0 numbers	`ATG/i`	ATG or atg
`\s+`	>= 1 space chars	`ATG/g`	all ATG's
`[^A]`	anything but 'A'	escaped characters: `\* \. \+ \\| \\ \/ \# \"`	
`\d{3}`	3 digit numbers		

Regex examples from parse_genbank.pl

- ORGANISM Mus musculus

```
if (/(ORGANISM\s*)(.*)/) { $org = $2; }
```

- VERSION NM_007553.1 GI:6680793

```
if (/(VERSION.*GI:)(\d*)/)  { $gi = $2;}
```

- CDS 357..1541

```
if (/(CDS\s*)(\d*)(\.\.)(\d*)/)
   {$start = $3; $end = $5; }
```

Hashes

- pairs of scalar data represented as a lookup table
- a hash can be created all at once:
  %hash = (key1, value1, key2, value2, etc.)
- examples: creating %translate and %gi

key	value
ATG ⇨	M
GGT ⇨	G
CAT ⇨	H
TAG ⇨	*

```
%translate = (
"ATG", "M",          "GGT", "G",
"CAT", "H",          "TAG", "*",
); # etc. . .

print "ATG is the codon for $translate{\"ATG\"}";
#      ATG is the codon for M
# In general, $hash{key} = value;
```


Hashes (cont.)

- a hash can also be created one key/value pair at a time:
  $hash{key} = value
- Example: given corresponding arrays of GI numbers (**@gi**) and sequences (**@seqs**), create **%gb**

```
for ($i = 0; $i <= $#seqs; $i++)
  {
      $gb{$gi[$i]} = $seq[$i];
  }
print "GI:$gi[$i] represents $gb{$gi[$i]}.";
# example:    GI:6680793 represents mouse BMP-2.
# To separate out keys and values:
@mykeys = keys(%gb); @myvalues = values(%gb);
```


Introduction to modules

- "a unit of software reuse"
- adds a collection of commands related to a specific task
- core modules vs. other modules
- see http://www.cpan.org/ to find documents and downloads, etc.

Using modules

- Before using a module that you installed yourself,

```
use lib 'full/path/to/module';
```

- For all modules,

```
use module_name;
```

- Example:

```
# full path to directory with GD.pm
use lib '/usr/people/elvis/modules';
use GD;       # The .pm is optional
```


Object-oriented Perl

- objects are module-specific references to data
- a module can describe multiple objects
  - Bio::SeqIO::fasta
  - Bio::SeqIO::GenBank
- **->** send information about the data
- example of creating an object and performing methods on it:

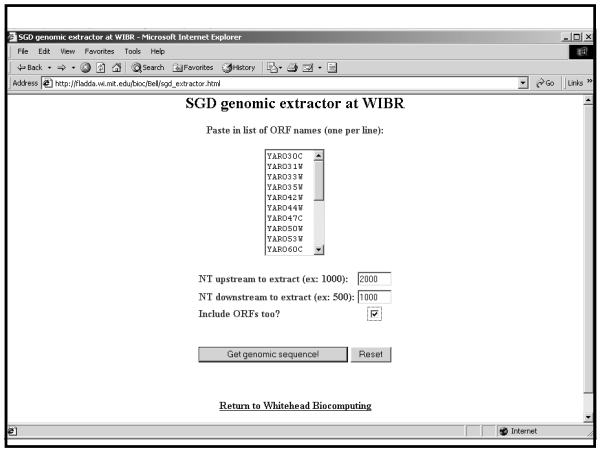
```
$myseqs = Bio::SeqIO->new(-file => "$inFile",
   '-format' => 'Fasta');      # makes a SeqIO object
$seqobj = $myseqs->next_seq(); # makes a Seq object
$rawseq = $seqobj->seq();
$rev_comp = $seqobj->revcom->seq();
```


LWP: fetch WWW documents

- To automate WWW access
- LWP::Simple - procedural interface to LWP
- Example of usage:

```
use LWP::Simple;
$url = "http://www.whatever.com/data.html";
$page = get($url);
if ($page)
   { # do something }
else    { print "Problems getting $url"; }
```

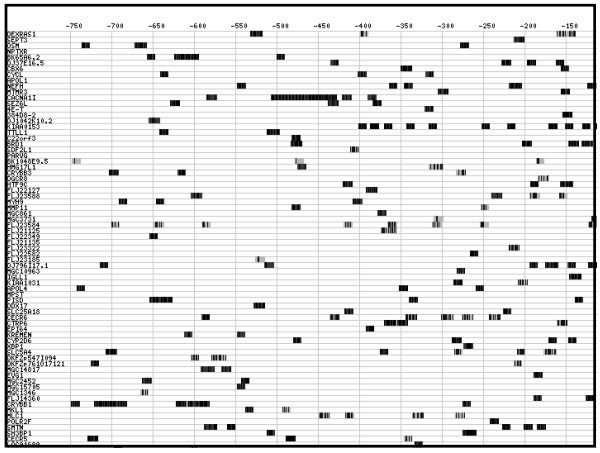
- example script: get_web_data.pl (NCBI queries)



CGI: run scripts from the WWW

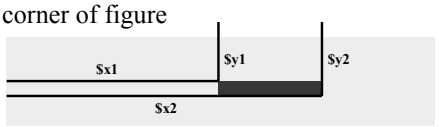
- gets input from HTML forms
- stdout writes document in browser
- execution controlled by server configuration
- example of usage:

```
use CGI qw(:standard);      # import :group shortcuts
$input = new CGI;
print $input->header('text/html');
# print content here
print $input->end_html;
```

GD: generate graphics

- GD generates figures (png, gif(?)) from rectanges, polygons, circles, lines, and text
- For all methods, position is in pixels from top left corner of figure



- method examples:

```
$img->filledRectangle($x1, $y1, $x2, $y2, $red);

$img->string(gdSmallFont, $x, $y, $text, $green);
```


BioPerl

- modules designed to simplify the writing of bioinformatics scripts
- uses objects (references to a specific data structure)
- Seq: main sequence object
  - available when a sequence file is read

```
$myseqs- = Bio::SeqIO->new('-file' =>
  "inputFileName", '-format' => 'Fasta');
$seqobj = $myseqs->next_seq();
```


BioPerl's SeqIO module

- sequence input/output
- formats: Fasta, EMBL, GenBank, swiss, SCF, PIR, GCG, raw
- parse GenBank sequence features
  - CDS, SNPs, Region, misc_feature, etc.
- sequence manipulation:
  - subsequence, translation, reverse complement

Using SeqIO

```
$in = Bio::SeqIO->new(-file => "$in", '-format' => 'Fasta');
$out = Bio::SeqIO->new(-file => ">$out", '-format' =>
   'Genbank');

while ($seq = $in->next_seq())
{
   $out->write_seq($seq);  # print sequence to $out
   print "Raw sequence:", $seqobj->seq();
   print "Sequence from 1 to 100: ", $seqobj->subseq(1,100);
   print "Type of sequence: ", $type = $seqobj->moltype();
   if ($type eq "dna")
   {
       $rev_comp = $seqobj->revcom->seq();
       print "Reverse complement: $rev_comp;
       print "Reverse complement from 1 to 100:",
             $seqobj->revcom->subseq(1, 100);
   }  }
```


BPlite: blast parser "lite"

- one of several blast parsers available
- Each matching sequence ("subject") can have multiple matching regions ("hsp", high scoring pair)

```
use Bio::Tools::BPlite;
$report = new Bio::Tools::BPlite(-file=>"$inFile");
while(my $sbjct = $report->nextSbjct)
{
  while (my $hsp = $sbjct->nextHSP)
    { print $hsp->subject->seqname; } }
```

- other blast parsers:
  - also for running BLAST and filtering results
  - examples: Bio::Tools::Blast , NHGRI::Blast

Bioinfo tools summary

- individual applications (Blast, Genscan, etc.):
  - web
  - command line
- analysis packages: EMBOSS, GCG, etc.
- Unix tools
- Perl tools
  - core commands
  - core modules
  - BioPerl and other "add on" modules

Project example: UTRs & translation

- get mRNA sequences (2 species)
- locate CDS
- extract 5' UTR and 3' UTR
- pattern matching
- pairwise alignment
- secondary structure
- graphical presentation of results
- describe new annotation to GenBank format

Demo

get_web_data.pl	use LWP to automate web file access
draw_figure.pl	draw a PNG figure using the GD module
fastaToGenbank.pl	sequence conversion
genbank_parse.pl	parse GenBank sequence features
manipulate_seq.pl	manipulate a sequence
blast_parse_1.pl	parse BLAST output files using BioPerl's BPlite
blast_parse_2.pl	parse BLAST output files using NHGRI's Blast