

Unix, Perl and BioPerl

III: Sequence Analysis with Perl - Modules and BioPerl

George Bell, Ph.D.
WIBR Bioinformatics and Research Computing

Unix, Perl, and BioPerl © Whitehead Institute, 2005

2

Sequence analysis with Perl Modules and BioPerl

- Regular expressions
- Hashes
- Using modules
- Library for WWW access in Perl (LWP)
- Common Gateway Interface Class (CGI)
- GD and SVG graphics libraries
- BioPerl (SeqIO, SearchIO)

Objectives

- Start to take advantage of the power of Perl's regular expressions
- Start to use modules to extend the power of Perl's core functions
- Start to use BioPerl modules for sequence analysis

Unix, Perl, and BioPerl © Whitehead Institute, 2005

3

Regular expressions

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

```
@hdrs = 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

Unix, Perl, and BioPerl © Whitehead Institute, 2005

4

Some uses of regular expressions

- biological applications you've seen:
 - sequence 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

Unix, Perl, and BioPerl © Whitehead Institute, 2005

5

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	\+ \ \\\ \/ \# \"	

Unix, Perl, and BioPerl © Whitehead Institute, 2005

6

Regex examples for GenBank files

- ORGANISM Mus musculus

```
if (/ (ORGANISM\s*) (.*) /)
{ $org = $2; }
```
- VERSION NM_007553.1 GI:6680793

```
if (/ (VERSION (.*) GI: (\d*) /)
{ $ver = $1; $gi = $2; }
```
- CDS 357..1541

```
if (/ (CDS\s*) (\d*) (\.\. (\d*) /)
{ $start = $2; $end = $4; }
```

Unix, Perl, and BioPerl © Whitehead Institute, 2005

7

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;
```

Unix, Perl, and BioPerl © Whitehead Institute, 2005

8

Hashes (cont.)

- a hash can also be created one key/value pair at a time:
\$hash{key} = value
- Example: given a file of GI numbers and sequence names
(ex: NM_001202 BMP4 ...), create %gi2seq

```
while (<IN>)
{
    @f = split ( / /, $ );
    $gi2seq{$f[0]} = $f[1];
    print "GI:$f[0] is $gi2seq{$f[0]}.";
}
# example:      GI: NM_001202 is BMP4.
# To separate out keys and values:
@mykeys = keys(%gi2seq);
@myvalues = values(%gi2seq);
```

Unix, Perl, and BioPerl © Whitehead Institute, 2005

9

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.

Unix, Perl, and BioPerl © Whitehead Institute, 2005

10

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 '/home/elvis/modules';
use GD; # The .pm is optional

Unix, Perl, and BioPerl © Whitehead Institute, 2005

11

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:

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

Unix, Perl, and BioPerl © Whitehead Institute, 2005

12

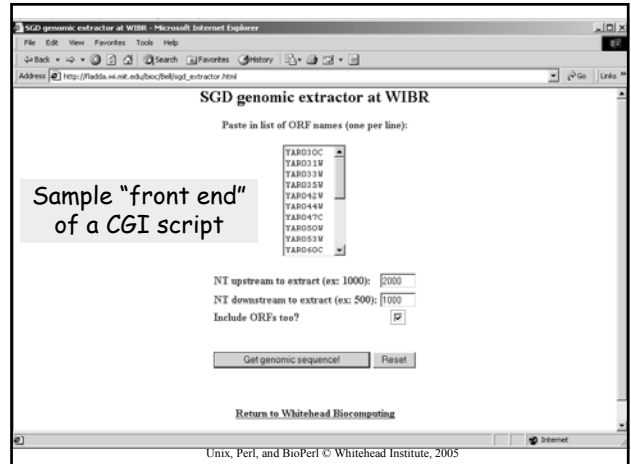
LWP: fetch WWW documents

- 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"; }
```

Unix, Perl, and BioPerl © Whitehead Institute, 2005

13



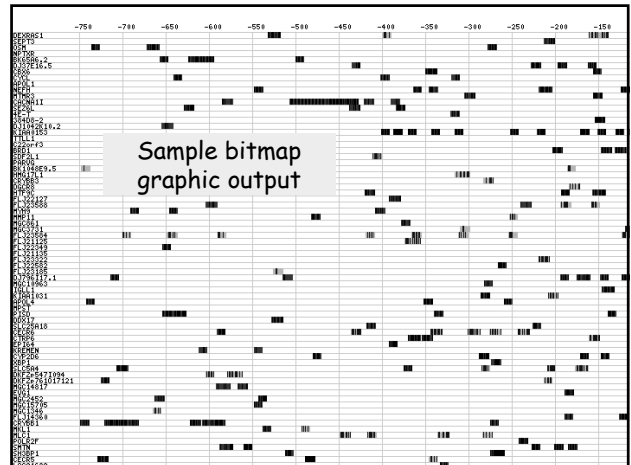
CGI: run scripts from the WWW

- 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;
```

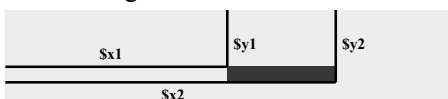
Unix, Perl, and BioPerl © Whitehead Institute, 2005

15



GD: generate bitmap graphics

- GD generates figures (png, gif(?)) from rectangles, 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);`

Unix, Perl, and BioPerl © Whitehead Institute, 2005

17

SVG: generate vector graphics

- Vector graphics
 - images are made up of objects
 - magnification maintains resolution
 - figures can be edited in Illustrator
- based on XML (text)
- SVG images can be viewed in a web browser BUT require a free plug-in

(<http://www.adobe.com/svg/>)

Unix, Perl, and BioPerl © Whitehead Institute, 2005

18

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

```
$seqs = Bio::SeqIO->new('-file' =>
"inputFileName", '-format' => 'Fasta');
$seqobj = $seqs->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 ($seqobj = $in->next_seq())
{
    $out->write_seq($seqobj); # print sequence to $out
    print "Raw sequence:", $seqobj->seq();
    print "Sequence from 1 to 100: ", $seqobj->subseq(1,100);
    print "Type of sequence: ", $type = $seqobj->alphabet();
    if ($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);
    }
}
```

Parsing BLAST reports with SearchIO

- best BioPerl blast parser

```
use Bio::SearchIO;
$report = new Bio::SearchIO(-file=>"$inFile",
-format => "blast");
while ($result = $report->next_result)
{
    while($hit = $result->next_hit)
    {
        while ($hsp = $hit->next_hsp)
        {
            print "Hit=", $hit->description, "\t",
"PercentID=", $hsp->percent_identity, "\n";
        }
    }
}
```

Summary: Perl and BioPerl

- Regular expressions
- Hashes
- Using modules
- Library for WWW access in Perl (LWP)
- Common Gateway Interface Class (CGI)
- GD and SVG graphics libraries
- BioPerl (SeqIO, BPlite)

Summary: Bioinformatics tools

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

Demo scripts on the web site

get_web_data.pl	use LWP to automate web file access
draw_figure.pl	draw a PNG figure using the GD module
draw_figure_SVG.pl	draw a figure with vector graphics
fastaToGenbank.pl	sequence conversion
genbank_parse.pl	parse GenBank sequence features
manipulate_seq.pl	manipulate a sequence
blast_parse.pl	parse BLAST output files using BioPerl's SearchIO

Exercises

- 1: Parsing a file of multiple BLAST reports
- 2: Manipulating a GenBank file with BioPerl and creating a PNG image
- 3: Setting up a web-based search script