Perl Programming Fundamentals for the Computational Biologist

Lab 1

Marine Biological Laboratory, Woods Hole
“Advances in Genome Technology and Bioinformatics”
Fall 2004

Andrew Tolonen
Chisholm Lab, MIT

Goals of lab 1: Let's say you are a computational biologist studying bacterial pathogenesis. You just returned from a seminar where the speaker described a novel amino acid motif found in genes involved infection. Your goal is to use Perl to search all the genes in your genome for this motif.

Prelab: download the genome file and the example solutions from a remove server

1. Open your browser

2. Type "ftp://PerlClass@cyano.mit.edu" into the browser window
   password: perlmbl

3. Retrieve the archive by "save target as" for Lab1.tar.gz

4. Unzip the archive

```
% gunzip Lab1.tar.gz
```

5. Untar the archive

```
% tar –xvf Lab1.tar
```

##

Exercise 1: Running a Perl script "helloWorld.plx"

open a file using a text editor (emacs, vi):

```
% emacs helloWorld.plx
```

write your script:

```
#!/usr/bin/perl

$message = "Hello world!\n";  #create a scalar to
                              #contain your message
```

```
print "$message";
```

close the file: emacs (Cntrl-X, Cntrl-S, Cntrl-X, Cntrl-C) or vi (Shift-ZZ)

make the script executable:
```
% chmod +x helloWorld.plx
```

run the script:
```
% ./helloWorld.plx
```

##

Exercise 2: Opening a Filehandle and reading a file into your program “openFile.plx”

1. Create a script to open a filehandle to the file “genes.faa”, filehandle syntax:

```
open(FILE, “<./myfile.txt”) or die “cant open
file\n”;
```

2. Read each line of the file into a scalar variable,

```
while ($line = <FILE>)
{
 $file = $file.$line;  # concatenate lines of the
                       # file into a variable
}
```

3. Print out the entire file to the screen.

##

Exercise 3: Feeding a file into a hash “makeHash.plx”

Ok. This is tough part. The goal here is to read the file and feed it into a hash (key = gene name; value = gene sequence).
As before, open the file and read each line. But this time you have to examine each line as the script reads it and either make a new key (line contains a gene name), or append the line to the gene sequence.

```
while ($line = <IN>)
{
 if (“line contains a gene name.  use a pattern
match here i.e. /gene/ or /^>/”)
 {
     “make a new hash key.  $key = $line;”
 }
 else
 {
   “concatenate line to genes sequence”
    $hash{$key} = $hash{$key}.$line;
 }
}
```

Here is a related example:

Problem: I have a file containing fly gene names and worm gene names.

```
CG6023
CG2012
F023.4
CG2323
```

I know that fly genes begin with “CG” and worm genes begin with “F”. I want to do two things: 1. separate worm and fly genes. 2. determine which genes occurred in the file more than once:

Solution: Open the file and read each line into a hash. If the gene name is new, create a key. If the gene name has occurred before, increment the value of the hash:

```
#!/usr/bin/perl
```

```
# count the number of duplicated gene entries in a
# file.  Fly gene names begin with “CG” and worm
# genes begin with “F”

open (IN, "<./genes.txt") or die "cant open input
file\n";

while ($line = <IN>)
{
 chomp $line;
 if ($line =~ /^CG/)
 {
   $fly{$line}++;
 }
 if ($line =~ /^F/)
 {
   $worm{$line}++;
 }
}

# print out the two hashes

foreach $k (keys(%fly))
{
 print "$k\$fly{$k}";
}


foreach $l (keys(%worm))
{
 print "$l\n$worm{$l}\n";
}

close IN;
close OUT;

############################################################
```

Exercise 4: Searching the values of a hash for a motif "searchHash.plx"

You've done the toughest part. Now you just want to search each sequence for your motif. You have a hash (key = gene name; value = gene sequence). Now just search each of the values in the hash for a motif.

```
$motif = "SG";  # motif is a serine followed by a
glycine

foreach $key (keys(%hash))
{
  if ($hash{$key} =~ /$motif/)
  {
    print "$key\n";
  }
}
```